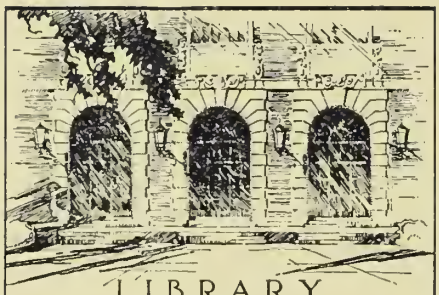
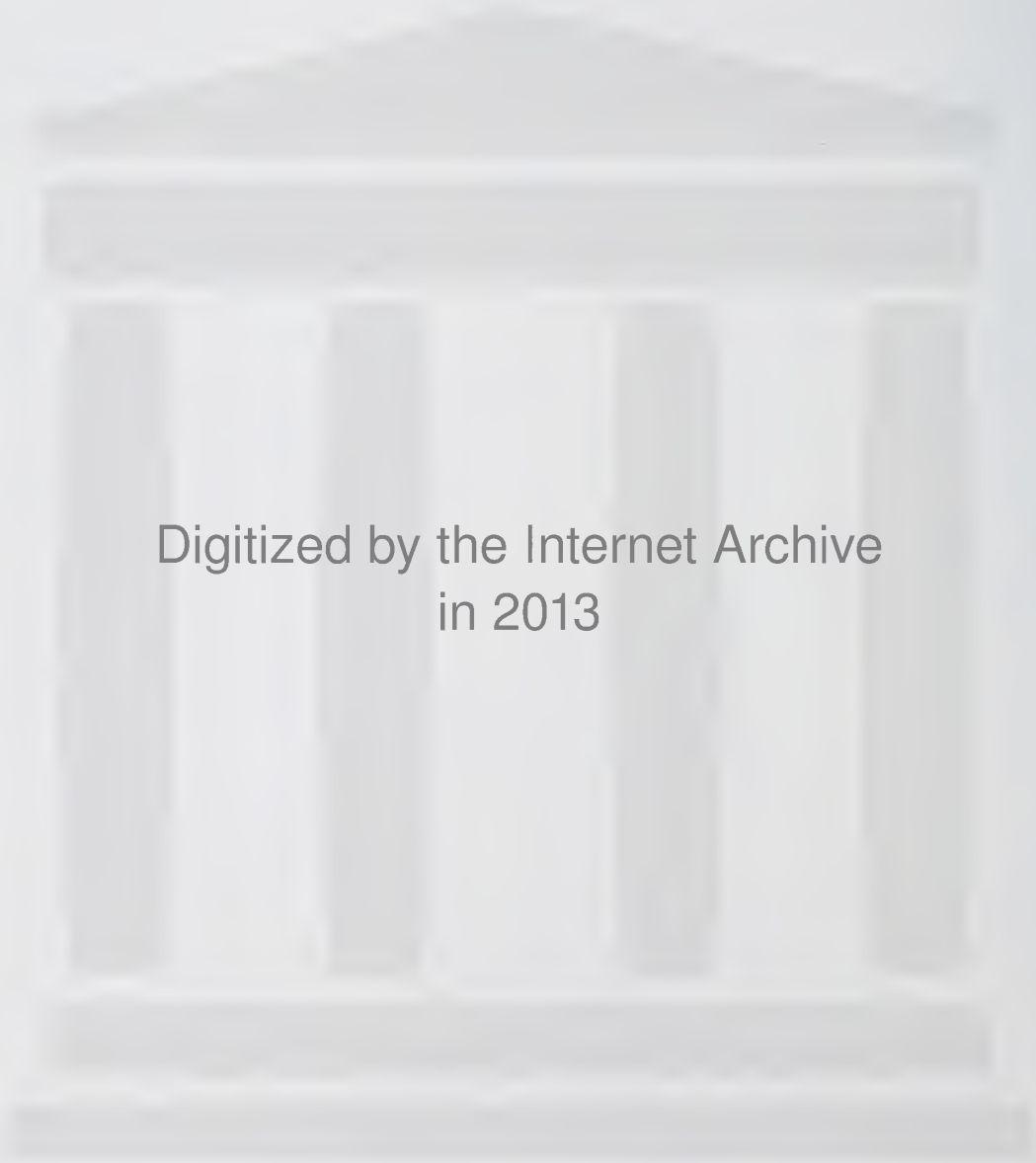




UNIVERSITY OF ILLINOIS
GRADUATE COLLEGE
DIGITAL COMPUTER LABORATORY

REPORT NO. 160

THE ARITHMETIC SUBSYSTEM OF THE NEW ILLINOIS COMPUTER

by

J. O. Penhollow

January 24, 1964

This work was supported in part by the
Atomic Energy Commission under Contract No. AT(11-1)-415

UNIVERSITY OF ILLINOIS
GRADUATE COLLEGE
DIGITAL COMPUTER LABORATORY

REPORT NO. 160

THE ARITHMETIC SUBSYSTEM OF THE NEW ILLINOIS COMPUTER

by

J. O. Penhollow

January 24, 1964

This work was supported in part by the
Atomic Energy Commission under Contract No. AT(11-1)-415

ACKNOWLEDGMENTS

The design and development of the Arithmetic Subsystem of the new Illinois computer was accomplished through the efforts of many people. To name all of them would be difficult, but a few deserve special recognition.

D. E. Muller, J. E. Robertson, D. B. Gillies, D. J. Wheeler, W. J. Poppelbaum, and G. Metze contributed many of the basic concepts which are incorporated in the present design.

The original set of logic circuits was developed by W. J. Poppelbaum and N. E. Wiseman. These circuits were later modified and augmented by J. E. Robertson, G. Metze, K. C. Smith, and H. Guckel.

H. Aiso, M. Faiman, R. R. Shively, R. E. Swartwout, and the author completed the final logic design under the competent and energetic leadership of D. B. Gillies. The work of this group was also directed and influenced by the advice and criticism of R. E. Meagher, J. E. Robertson, and D. E. Muller. A number of errors in the final design of delayed control were discovered and corrected by R. H. Farrell.

In the development stage, the design group benefited from the counsel and guidance of J. E. Robertson, C. E. Carter, H. E. Lopeman, and T. E. Kerkering. The set of logic drawings which describe the subsystem was produced by H. Aiso, R. E. Swartwout and the author with the help of H. E. Lopeman, S. P. Krabbe, R. F. Kingsley, K. C. Law, J. K. Burrell, and other members of the drafting department. The component layout was accomplished by R. L. Cummings, M. D. Freedman, R. F. Kingsley, S. P. Krabbe, and L. J. Peck under the direction of H. E. Lopeman and A. F. Irwin. T. E. Kerkering and F. P. Serio supervised the construction.

TABLE OF CONTENTS

		Page
1.	INTRODUCTION	1
2.	ORGANIZATION OF THE ARITHMETIC SUBSYSTEM AND ITS RELATIONSHIP WITH THE EXECUTIVE SUBSYSTEM	6
3.	THE ARITHMETIC UNIT	10
3.1	The Main Arithmetic Unit	10
3.1.1	A Summary of the MAU Logic	16
3.1.2	The MAU Bit Path Logic	21
3.1.2.1	The F1 to M Path	22
3.1.2.2	The R to M Path	22
3.1.2.3	The M Register	22
3.1.2.4	The MsA Selector	23
3.1.2.5	The MsS Selector	24
3.1.2.6	The sA Selector	24
3.1.2.7	The A Register	30
3.1.2.8	The sS Selector	31
3.1.2.9	The S Register	37
3.1.2.10	The sQ Selector	37
3.1.2.11	The Q Register	39
3.1.2.12	The sR Selector	40
3.1.2.13	The R Register	42
3.1.2.14	The R to FO Path	42
3.1.3	Standard Base 4 Pseudo Adders	45
3.1.4	High Special Adders	48
3.1.5	Low Special Adders	52
3.1.6	One Digit Assimilators	57
3.1.7	Zero Detectors and Round-Off Logic	58
3.1.8	Carry Generator	60
3.1.9	The α Logic	65
3.1.10	The A, S, M and R Normalization Logic	67
3.1.11	Sign A and Sign Comparison Logic	69
3.1.12	The Q and R Half-Subtractors	70
3.1.13	The Carry-Borrow Logic	74
3.1.14	The θ Decoder	78
3.1.15	The μ Decoder	79
3.1.16	The Division Predictors (ρ Decoders)	87
3.1.17	The Quotient "Bit" Recoder and the Borrow-Subtractor	106
3.2	The Exponent Arithmetic Unit	116
3.2.1	A Summary of the EAU Logic	120
3.2.2	The EAU Bit Path Logic	120
3.2.2.1	The F1 to EM Path	120
3.2.2.2	The EM Register and the EM = -64 Logic	121
3.2.2.3	The sD Selector	121
3.2.2.4	The sEA Selector	123
3.2.2.5	The EA Register	123
3.2.2.6	The ES Register and the ES to FO Path	124
3.2.2.7	The sE Selector	124
3.2.2.8	The E Register	125

Page

3.2.3 The Exponent Adder (D-Adder) 126
3.2.4 The Exponent Decoder 128

4. THE LINK MECHANISMS. 135

4.1 Gates 137
4.2 Selector Mechanisms 140
4.3 Control Status Memory Elements. 148

5. DELAYED CONTROL. 154

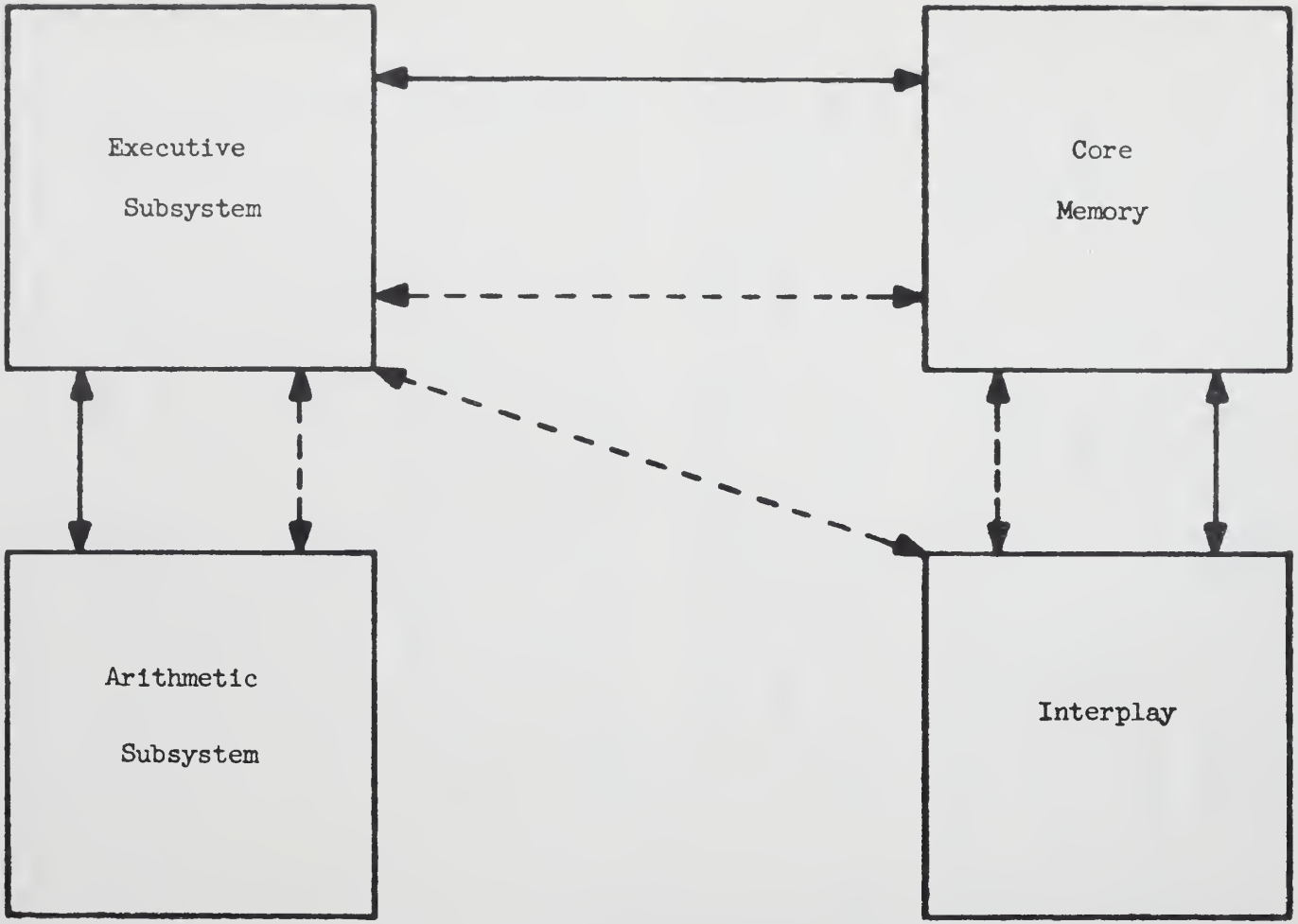
5.1 The Decode Sequence G 161
5.2 The Clear Add Sequence G 174
5.3 The Correct Overflow and Detect Zero Sequence K 179
5.4 The Load Q Sequence L 183
5.5 The Add Sequence A 185
5.6 The Exponent Arithmetic Sequence E 201
5.7 The Store Sequence S 205
5.8 The Shift Sequence F 212
5.9 The Store Preliminaries Sequence P 217
5.10 The Normalize Sequence R 219
5.11 The Difference Absolute Value Sequence V 221
5.12 The Multiply Sequence M 222
5.13 The Division Sequence D 226

1. INTRODUCTION

The new Illinois computer is composed of four major subsystems as shown in Fig. 1. The Executive Subsystem includes Advanced Control (AC) with its counters and interlocks, the Address Arithmetic Unit (AAU), and the Flow Gating Memory (FG). The Arithmetic Subsystem contains Delayed Control (DC), the Link Mechanisms (LM), and the Arithmetic Unit (AU). The Core Memory Subsystem presently contains a 4096 word Core Memory (C) with its control, selection, and buffering logic. The Interplay Subsystem includes auxiliary storage, I/O devices, and the associated control logic.

This report presents a comprehensive description of the logical structure and function of the Arithmetic Subsystem. It is written primarily as a reference for those with maintenance responsibilities. As such, it does not dwell on the underlying philosophy.

The reader is assumed to be familiar with the Delayed Control order code as defined in File No. 458, "Order Code for the New Illinois Computer," by D. B. Gillies. It is recommended that the reader obtain the set of drawings entitled "Logic Drawings for the Arithmetic Subsystem." In the discussion of the subsystem logic, frequent reference is made to this set of drawings, and in particular to the drawing (D-1128) entitled "Composite Flow Chart for all MAU and EAU Arithmetic Operations." Throughout the report, the drawings included in this set are referred to by number; e.g., D-1128. The following reports are recommended as references: File No. 319, "Theory of Computer Arithmetic Employed in the Design of the New Computer at the University of Illinois," by J. E. Robertson: File No. 397, "A Description of the Operation of Delayed Control in Terms of Its Flow Charts," by D. B Gillies; File No. 388, "One Method for Designing Speed-Independent Logic for a Control," by R. E. Swartwout; and File No. 528, "A Description of the Logic Drawings for the Arithmetic Subsystem," by J. O. Penhollow.



Control Path - - - - -

Data Path ——————

Figure 1. General Organization of the New Illinois Computer

The nomenclature used in this report agrees with that shown on the "Logic Drawings for the Arithmetic Subsystem," except that suffixes on signal names are usually omitted if their purpose is to distinguish signals which are logically identical but electronically different. In line with this policy, no special attempt is made to explain the existence of logical identity elements such as emitter followers and noninverting amplifiers or cable drivers.

Throughout the report, "1" and "0" denote positive and negative voltage levels respectively. If a memory element such as Z is set to "1," its true or "1" side output -- designated z -- has a logic value of "1," meaning the z signal voltage is positive with respect to ground. In this state, the false or "0" side output -- designated $\bar{z}$ -- has a logic value of "0" meaning the $\bar{z}$ signal voltage is negative with respect to ground. If Z is set to "0," z and $\bar{z}$ have logic values of "0" and "1" respectively. The parity of the signal used to set a memory element to a particular state depends on the type of memory element under consideration. It is worthwhile to note that memory elements and their inputs are generally designated by capital letters, while their outputs are designated by the same letters in lower case. On this and other questions of nomenclature, the reader is referred to File No. 528 as indicated above.

The symbolism used to denote the thirteen basic control sequences which form Delayed Control is defined at the beginning of section 5 of this report. For example, A denotes the floating add sequence, and S denotes the store sequence. This notation agrees with that shown on the "Composite Flow Chart for all MAU and EAU Operations" (D-1128).

A word of caution is in order concerning the Boolean expressions which define conditional branching on the composite flow chart. Control takes a particular branch if the corresponding Boolean expression has a logic value of "1" or equivalently if its dual has a logic value of "0." The Boolean expressions which appear on the composite flow chart are considered positive logic and their

duals negative logic. Due to the nature of the control circuitry, the branch conditions are realized with negative logic in the machine, and they appear in this form on the control logic drawings.

The report begins with a general description of the organization of the Arithmetic Subsystem and its relationship to the Executive Subsystem in section 2. No attempt is made to evaluate the merits of this relationship.

Section 3 contains a lengthy discussion of the Arithmetic Unit which includes the Main Arithmetic Unit (MAU), the Exponent Arithmetic Unit (EAU), and all encoders or decoders which either feed or are fed by the registers, selectors, and adders of the MAU and EAU. The Boolean expressions for each unit of logic in the MAU and EAU are defined and discussed. The purpose and function of each unit is indicated with reference to the DC order code. In certain cases the logic is analyzed to provide additional insight.

It should be kept in mind that section 3 treats each unit of logic in the Arithmetic Unit as a separate entity. A particular unit may be used during the execution of many different DC orders; however, the purpose it serves may not be the same in all cases. In an attempt to illustrate this fact, the purpose of each unit is described relative to all of the DC orders which use it. This description generally does not include the operational details of the corresponding control sequences. However, the half-subtractors, multiplier recoders, and division predictors are so specialized that their description includes some of these details for reasons of clarity.

The Link Mechanisms (LM) are discussed in section 4. These include gates, selector mechanisms, and status memory elements. Delayed Control governs the data flow in the Arithmetic Unit via these devices. The operation of each type of Link Mechanism is described in this context.

Section 5 describes Delayed Control in terms of its thirteen basic sequences. The discussion is supplemented with pertinent references to those

sections which describe the Arithmetic Unit and the Link Mechanisms. A description of the signal flow through the decode G , clear add B , and correct overflow K sequences is presented to illustrate the characteristics of speed-independent control logic. The reader is referred to File No. 388 by R. E. Swartwout for a more thorough treatment of this topic.

2. ORGANIZATION OF THE ARITHMETIC SUBSYSTEM AND ITS RELATIONSHIP WITH THE EXECUTIVE SUBSYSTEM

A block diagram of the Arithmetic Subsystem is shown on the right half of Fig. 2. The subsystem is capable of performing base 4 floating point arithmetic and a limited set of logical operations as described in File No. 458 by D. B. Gillies. The input and output operand channels carry 52 bits in parallel. The first 45 bits of the operand are generally interpreted as a fraction in the range: $-1 \leq f < 1$. The last 7 bits are generally interpreted as an integer base 4 exponent in the range: $-64 \leq x < 64$. Both the fraction and the exponent have a complement representation. The other data channel between the Executive and Arithmetic Subsystems carries a 6 bit Delayed Control (DC) order which specifies the operation performed by the Arithmetic Subsystem.

As shown in Fig. 2, the Arithmetic Subsystem is composed of three principal units. The Arithmetic Unit (AU) contains the computational logic and is divided into two major subunits as indicated. The Main Arithmetic Unit (MAU) and the Exponent Arithmetic Unit (EAU) handle the fractional and exponential calculations respectively. The second principal unit of the subsystem contains the Link Mechanism (LM) logic. This logic transmits commands from the Delayed Control (DC) to the Arithmetic Unit (AU). It may be further subdivided into gate and selector mechanisms and status memory elements. Delayed Control (DC) is the third principal unit of the Arithmetic Subsystem. The DC logic governs the data flow in the AU via the LM.

A block diagram of the Executive Subsystem is shown on the left half of Fig. 2. This subsystem supervises the execution of the stored program. The other three subsystems are subordinate in the sense that they receive their orders from the Executive Subsystem. The Executive Subsystem transfers command words of the stored program from Core to the Flow Gating Memory, where it sequentially examines the individual orders. It executes an order or portions of an order

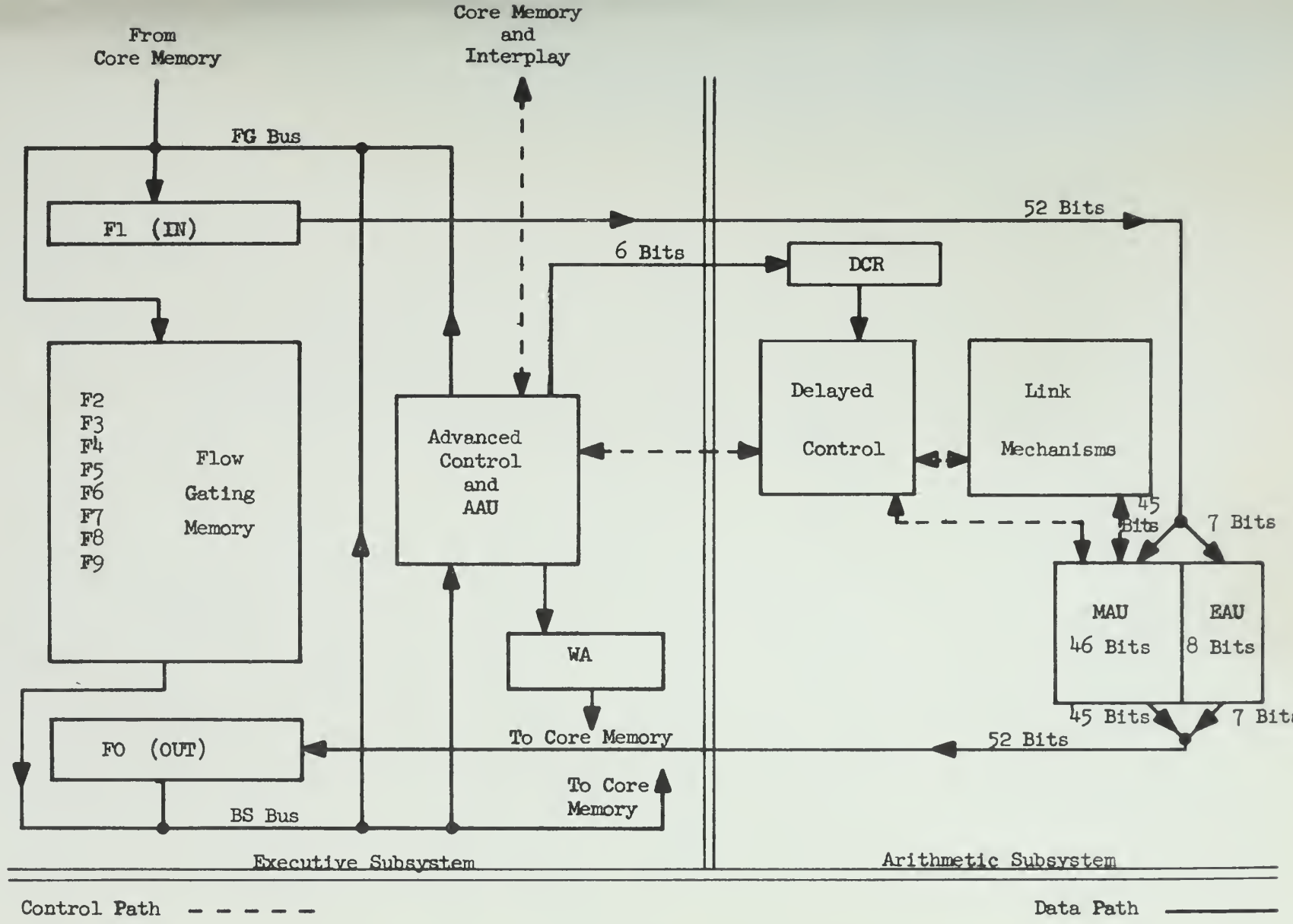


Figure 2. Control and Data Flow between the Executive and Arithmetic Subsystems

for which it has the capability. Otherwise, it assigns all or part of the order to an appropriate subsystem. In the latter case, the Executive Subsystem may send and/or receive a corresponding operand. Address modification, indexing, and certain control transfer orders are executed within the Executive Subsystem. All orders requiring the transfer of operands between Flow Gating Memory and Core are partially or totally executed by the Executive Subsystem.

To provide a background for the detailed description of the Arithmetic Subsystem, we briefly consider the functional relationship between it and the Executive Subsystem. The reader is referred to File No. 464, "Advanced Control," by R. R. Shively for a comprehensive discussion of the purpose and function of the Executive Subsystem and its relationship with the Arithmetic Subsystem.

When Advanced Control (AC) of the Executive Subsystem encounters an order which must be assigned to Delayed Control (DC) of the Arithmetic Subsystem, it first decides whether the order presently held in the Delayed Control Order Register (DCR) has been decoded and initiated by DC. If so, AC transfers the new order to DCR. If not, AC may have to wait until DC completes its present order and decodes the order in DCR. If the order requires an initial operand, AC determines whether DC has used the operand presently held in the IN register (i.e., the F1 register of the Flow Gating Memory). If so, AC places the new operand in IN; otherwise, it must wait unless there are other orders which it may perform or assign. If the order requires a terminal operand (i.e., a DC store order), AC checks the contents of the OUT register (i.e., the FO register of the Flow Gating Memory) and its Core address in the WA register. As long as no initial operand is required, DC is permitted to initiate the store order even though the Executive Subsystem has not transferred the present contents of OUT to the Core address held by WA. However, DC cannot complete the store order until this transfer is made. As soon as Out Write Control (a part

of the Executive Subsystem) has transferred the present contents of OUT to Core, AC places the new Core address in WA. DC is then free to gate the terminal operand from the AU into OUT and thus complete its part of the store order.

There are four special DC store orders which have initial and terminal operands with the same Core address. These are ASC, SSC, XCH, and SEQ. When AC encounters one of these orders, it must check and load DCR and IN under the conditions given above. If the present contents of OUT have been stored in Core, the new Core address is placed in WA and DC is permitted to gate the terminal operand into OUT.

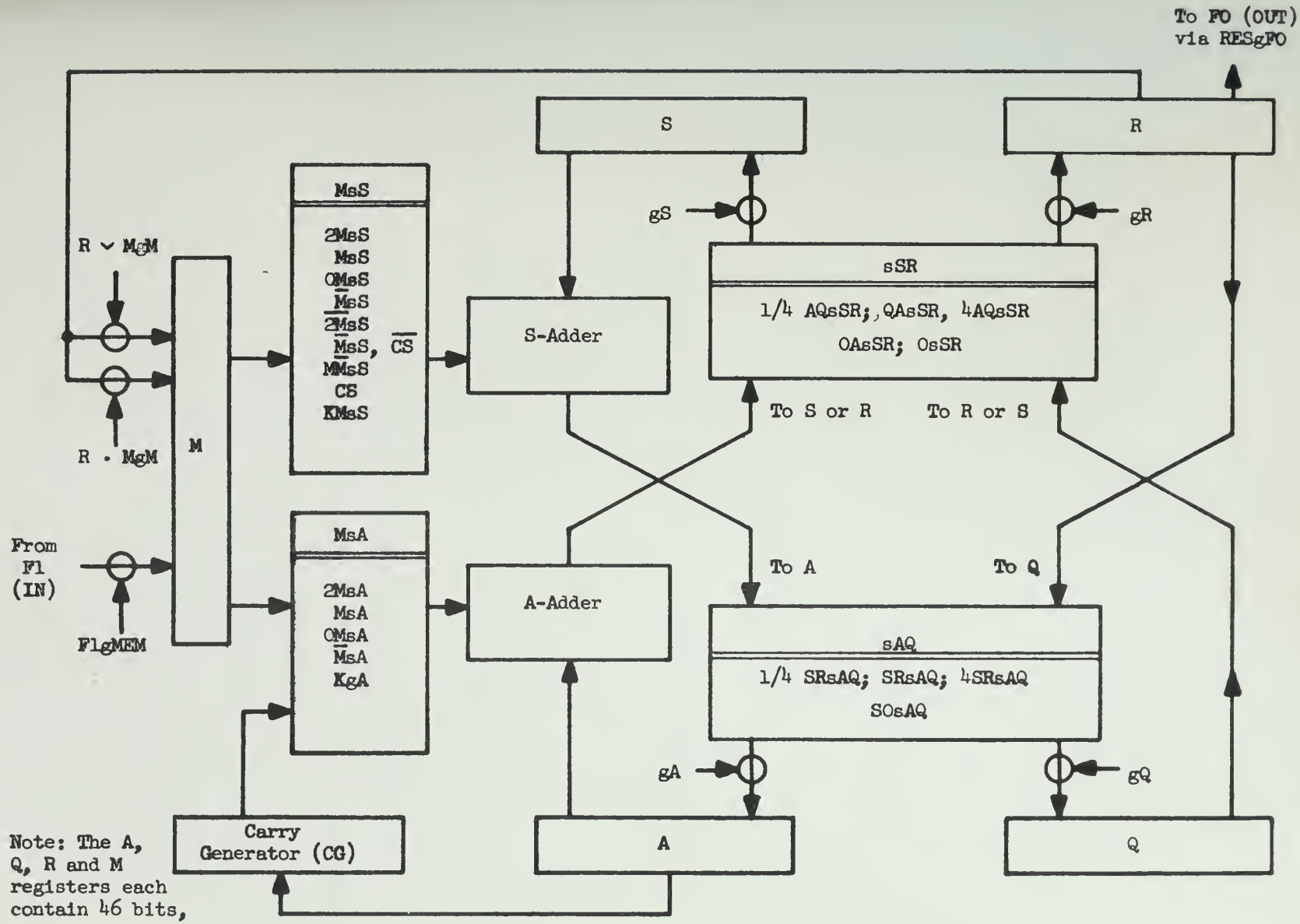
In summary, the Executive and Arithmetic Subsystems operate concurrently but not in a completely independent manner. The former assigns orders and operands to the latter and accepts operands as a result. The Executive Subsystem has no control over the time or equipment used by the Arithmetic Subsystem to obtain a result. With few exceptions, the Executive Subsystem is capable of assigning a new order to the Arithmetic Subsystem before the latter has completed its current order. As implied above, there are circumstances in which the Arithmetic Subsystem is forced to wait on the Executive Subsystem. However, these circumstances can either be avoided or minimized by careful programming.

3. THE ARITHMETIC UNIT

The Arithmetic Unit (AU) consists of the Main Arithmetic Unit (MAU) and the Exponent Arithmetic Unit (EAU). These two units operate concurrently, but are physically and logically distinct. Both receive their operands from the 52 bit IN register. The first 45 bits of this are usually interpreted as a fraction, $-1 \leq f < 1$, and is the MAU operand. The last 7 bits are usually interpreted as an exponent, $-64 \leq x < 64$, and is the EAU operand. The complete floating point operand contained by IN may be expressed as $p = f \cdot 4^x$. Floating point results placed in OUT have the same form. Both f and x are in complement representation. In the case of logical orders other than SRS and LRS, x is not used by the EAU, and the MAU interprets f as a 45 bit word without sign. The same is true for the orders SRS and LRS except that x is used by the EAU to govern the number of base 4 shifts. In strictly exponent operations the MAU is not active and thus does not use f. Neither f nor x are used during the execution of store orders which do not require an initial operand.

3.1 The Main Arithmetic Unit

The general topology of the MAU is shown in Fig. 3. Registers A, M, Q and R each have 46 bits, while S has 48 bits. Since the two adders yield sums in base 4 stored carry representation, A and S also contain 23 and 24 stored carry bits respectively. Three stored carry bits are also associated with the Q and R registers to facilitate the performance of certain additions, multiplication, and division. With regard to these special bits, the term "stored carry" is inaccurate, since some of them have negative weight. The bits associated with each of the five registers are indicated below. Stored carry bits are designated with an asterisk.



Note: The A, Q, R and M registers each contain 46 bits, where the S register contains 48 bits.

Figure 3. Block Diagram of the Main Arithmetic Unit

A: A^*_0 A^*_2 A^*_i A^*_{44}

A_{-1} A_0 A_1 A_2 A_{i-1} A_i A_{43} A_{44}

S: S^*_{-2} S^*_0 S^*_2 S^*_i S^*_{44}

S_{-3} S_{-2} S_{-1} S_0 S_1 S_2 S_{i-1} S_i S_{43} S_{44}

M: M_{-1} M_0 M_1 M_2 M_i M_{43} M_{44}

Q: Q^*_0 Q^*_{42} Q^*_{44}

Q_{-1} Q_0 Q_1 Q_2 Q_i Q_{41} Q_{42} Q_{43} Q_{44}

R: R^*_0 R^*_{42} R^*_{44}

R_{-1} R_0 R_1 R_2 R_i R_{41} R_{42} R_{43} R_{44}

The MAU does modulo 4 arithmetic. The weight of a bit in the $i^{\underline{th}}$ position (stored carry or otherwise) is 2^{-i}. The point is assumed to lie between the $0^{\underline{th}}$ and $1^{\underline{st}}$ bit position, so the largest range of f that can be associated with every register of the MAU is $-2 \leq f < 2$. If the bits S_{-3}, S_{-2} and S^*_{-2} are ignored, the positive limit on f in the A and S registers is less than 8/3. The limit, 8/3, is obtained by assuming an infinite stored carry representation.

The function of each register in the MAU is briefly described below. During the decode step (G1) of every DC order, the gate FlgMEM transfers the

first 45 bits of IN to M even though the order does not use an initial operand. The IN_0 bit is mapped into M_{-1} and M_0. The results of the previous DC operation are generally held in A and Q which represent the primary rank of the double length accumulator. The S and R registers form the secondary rank of the double length accumulator which usually holds an intermediate result at the end of a DC operation. An exception occurs in the case of divide orders. At the end of any divide order, A contains the fractional quotient, Q contains zeros, and R contains the fractional remainder. During the store step (S9) of every DC store order, the RESgFO gate transfers a modified copy of R to the OUT register. The nature of the modification depends on the store order and will be discussed in detail later. It is immediately clear, however, that a bit for bit transfer is impossible since the information contained in 46 bits of R must be stored in the first 45 bits of OUT.

The two adders shown in Fig. 3 are composed of base 4 pseudo adder modules which are standard except at the most and least significant ends of the A and S registers. The A adder has the contents of the A register as one input and the output of the MsA selector as the other. In either case, the selector output in two's complement representation is added to the stored carry representation held in A or S. A subtraction is accomplished by causing $\overline{M}$ to appear at the selector output and then adding an extra bit, ca or cs, in the $44^{\underline{th}}$ position. The outputs of the A and S adders are designated as α and σ respectively, and are generally in stored carry representation. Let a, s, and m represent the contents of A, S, and M. The values of α and σ in terms of a, s, and m under various M selector settings are given below. Since the M selector mechanisms have memory, the choice of a particular setting remains in effect until a new choice is made.

2 MsA:	$\alpha = a + 2m$	
MsA:	$\alpha = a + m$	
0MsA:	$\alpha = a + 0$	The representation of a and α may not be the same since the adder propagates the stored carries of "a" over one base 4 position.
MsA:	$\alpha = a - m$	
KgA:	$\alpha = a + k$	In this notation, k denotes the output of the carry generator. As a result, α is the assimilated form of a; i.e., $\alpha_i^* = 0$ for all i.
2MsS:	$= s + 2m$	
MsS:	$= s + m$	
0MsS:	$= s + 0$	The comment concerning $\alpha = a + 0$ also applies here. As a consequence of the nature of the MsS selector mechanism, the 0MsS setting can only be used following a setting of 2MsS, MsS, $\bar{M}$sS. Otherwise KMsS must be used.
CS:	$= s + 2^{-44}$	
$\bar{M}$MsS:	$= s - 2^{-44}$	
KMsS:	$= s + 0$	This setting is used following a setting of CS or $\bar{M}$MsS.

The fact that stored carries are permitted in A and S but not elsewhere, implies they must be assimilated whenever transfer to another register is necessary. The purpose of the carry generator logic is to permit the assimilation of all stored carries in A by one pass through the A adder. The carry generator was first proposed and described by D. J. Wheeler in Report No. 92 entitled "The Arithmetic Unit." In effect, it generates a carry word which may have "1's" in the even bit positions and always has "0's" in the odd bit positions. An even bit position will contain a "1" if the assimilation of all base 4 stored carries in A will cause a carry into that position.

The addition of this carry word to the contents of A yields an assimilated representation of A, provided the generated stored carries are suppressed.

The sAQ and sSR selector mechanisms also have memory. Once a particular selector setting has been chosen by DC, it remains in effect until a new setting is made. The settings shown in Fig. 3 are easily interpreted, provided the outputs of the A and S adders are used in place of the register outputs. The effect of these settings is indicated below. The presence of gA, gQ, gS and gR above the arrows implies that these gates must be active before the selector outputs can be transferred to the associated registers.

$$1/4\ \text{AQsSR:}\quad 1/4\ \alpha \overset{gS}{\rightarrow} S,\ 1/4\ Q \overset{gR}{\rightarrow} R$$

$$\text{QAsSR:}\quad Q \overset{gS}{\rightarrow} S,\ \alpha \overset{gR}{\rightarrow} R$$

$$4\ \text{AQsSR:}\quad 4\ \alpha \overset{gS}{\rightarrow} S,\ 4\ Q \overset{gR}{\rightarrow} R$$

$$\text{OAsSR:}\quad 0 \overset{gS}{\rightarrow} S,\ \alpha \overset{gR}{\rightarrow} R$$

$$\text{OsSR:}\quad 0 \overset{gS}{\rightarrow} S,\ 0 \overset{gR}{\rightarrow} R$$

$$1/4\ \text{SRsAQ:}\quad 1/4\ \sigma \overset{gA}{\rightarrow} A,\ 1/4\ R \overset{gQ}{\rightarrow} Q$$

$$\text{SRsAQ:}\quad \sigma \overset{gA}{\rightarrow} A,\ R \overset{gQ}{\rightarrow} Q$$

$$4\ \text{SRsAQ:}\quad 4\ \sigma \overset{gA}{\rightarrow} A,\ 4\ R \overset{gQ}{\rightarrow} Q$$

$$\text{SOsAQ:}\quad \sigma \overset{gA}{\rightarrow} A,\ 0 \overset{gQ}{\rightarrow} Q$$

Since Q and R represent extensions of the A and S registers respectively, 1/4 AQsSR and gR causes α_{43}, α_{44}, and α^{*}_{44} to be mapped into R_{-1}, R_0, and R^{*}_0. Likewise, 4 SRsAQ and gA causes r_{-1} (ρ_{-1}) and r_0 (ρ_0) to be mapped into A_{43} and A_{44}. Similar interpretations must be made for the other options. A detailed discussion of the logical connections at the ends of the A, S, Q and R registers is given in section 3.1.2.

The gate mechanisms do not have memory, so DC must activate them each time the contents of the associated registers are changed. If the gate is not activated, the register simply retains its old contents regardless of the bit

configuration appearing at its inputs. The two special gates which have not been mentioned so far are R · MgM and R ∨ MgM. Their notation suggests their action. R · MgM transfers the zeros of R into M. The bitwise AND of the previous contents of M and the contents of R is left in M. Likewise, R ∨ MgM transfers the ones of R into M, yielding the bitwise OR of M and R. When R · MgM and R ∨ MgM are active together, the contents of R are simply transferred to M.

3.1.1 A Summary of the MAU Logic

The logic associated with the MAU is completely illustrated in a series of 15 drawings. These drawings are part of the set of drawings entitled "Logic Drawings for the Arithmetic Subsystem." A summary of the logic shown on each of these drawings follows.

"Standard A, S and M Logic" (SAS) D-1707 illustrates a standard 4-bit cross section of the A, S and M registers with their associated gates and selectors. The standard base 4 pseudo adders are shown in block form. This logic is physically located on the Q, A and S chassis of a standard MAU bay, e.g. 3F or 3R.

"High A, S and M Logic" (HAS) D-1522 describes the logic at the most significant ends of the A, S and M registers down to and including bits A_4, S_4, and M_4. Gates, selectors, and some of the high A and S end connection logic appear in detail on this drawing. The standard base 4 pseudo adders, and the high A and S special adders are shown in block form. The logic shown on this drawing is physically located on chassis Q7F, Q6F, A8F, A7F, A6F, S8F, S7F and S6F.

"Low A, S and M Logic" (LAS) D-1523 contains the logic at the least significant ends of the A, S and M registers up to and including bits A^*_{40},

S^*_{40} and M_{40}. Gates, selectors and some of the low A and S end connection logic are shown. The standard base 4 pseudo adders and the low A and S special adders appear in block form. This logic is physically located on chassis Q5R, A6R, A5R, S6R and S5R.

"Standard A-Adder Logic" (STA) D-1265 includes two standard base 4 pseudo A-adders with their carry inhibit (KI) gate. These adders were not realized with basic logic elements throughout. Consequently, open arrowhead inputs to AND and AND-OR complexes denotes diodes instead of transistors. The logic shown on this drawing is located on the A chassis of a standard MAU bay.

"Standard S-Adder Logic" (STS) D-1266 shows two standard base 4 pseudo S-adders. The comments concerning the logical symbolism on the STA drawing also apply here. The adders are physically located on the S chassis of a standard MAU bay.

"High A and S Special Adder Logic" (HAD) D-1526 illustrates the special adders at the most significant end of the A and S registers. Generally speaking, these adders are special variations of the standard base 4 pseudo adders. Carry signals (c_i, d_i, x_i, y_i) to the next higher base 4 pseudo adders are not needed and therefore are not generated. Because of the fan-out required for the division predictors, α-logic and circular shift connections, the outputs contain amplifiers and cable drivers that are associated with the standard adder outputs. The high S adder also provides a partial sum over the bits S_{-3}, S_{-2}, and S^*_{-2}. Chassis A7F and S8F contain the high A-adder logic, while chassis S7F and A8F contain the high S-adder logic.

"Low A and S Special Adder Logic" (LAD) D-1212, shows the special adders at the least significant end of the A and S registers. These adders are considerably different than the standard adders. They were realized with basic logical elements throughout, and must generate two stored carries as opposed to

one stored carry for a standard adder. The latter requirement is a consequence of the fact that the complement carry (ca, cs) appears as a unit in the 44th position. This means that four units may have to be added in the 44th position without violating the adder relation. The significance of this will be discussed later. The low A adder must also send information to the round-off logic and multiplier recoder logic. Chassis A6R and A5R contain the low A-adder logic, while chassis S6R and S5R contain the low S-adder logic.

"Carry Generator Logic" (CG) D-1099 contains the logic used during the assimilation of all stored carries in A. The carry generator has the bits of A and the round-off, ρ, as inputs. It produces a carry word as described earlier. When the carry word is added to A via KgA, the output of the A-adder, α, is the assimilated form of A. The stored carries α_i^* which are generated during this addition are forced to zero by the carry inhibit signal, $\overline{KI}$, which is 0 (negative voltage) during the time KgA is active (i.e., KgA = 1 so that the carry word is added to A in the A-adder). The carry generator logic is distributed throughout the A chassis level of the MAU as indicated by the drawing.

"Standard Q and R Logic" (SQR) D-1706 contains a standard 4-bit cross section of the Q and R registers with their associated gates and selectors. R $\cdot$ MgM and R $\vee$ MgM gate logic also appears. The Q level chassis of a standard MAU bay contains the logic shown on this drawing.

"High Q and R Logic" (HQR) D-1525 illustrates the logic at the most significant end of the Q and R registers down to and including bits Q_4 and R_4. In addition to the standard gate and selector logic, most of the high Q and R end connection logic appears on this drawing. The latter includes the Q and R one digit assimilators and the logic which modifies the contents of R on the way to OUT during S9 of the store sequence. The logic shown on this drawing is physically located on chassis Q7F and Q6F.

"Low Q and R Logic" (LQR) D-1524 shows the logic associated with the least significant end of the Q and R registers up to and including bits Q_{37} and R_{37}. A large part of the low Q and R end connection logic as well as the standard gate and selector logic appears on this drawing. The borrow subtractor logic which is used during the generation of quotients is shown in block form. Part of the logic which forms the inputs to Q^*_{42} and R^*_{42} is shown. These bits are mode bits during multiplication, but act as carry and borrow bits respectively during division. The logic used to overwrite the bits in R_{39}, R_{40}, R_{41}, R_{42}, R_{43} and R_{44} with the bit in ES_7 during the RESgFO gate of an SEX order also appears on this drawing. The logic shown on this drawing is physically located on chassis Q4R, Q5R, Q6R, B4C, B5C, and B6C.

The "μ and θ Decoder Logic" ($\mu\theta$D) D-1506 contains the decoder logic which is used to set the MsA and MsS selectors during multiplication and addition. The μ decoder is used during multiplication only. The outputs of the μMsA logic set the MsA selector to either 2MsA, MsA, 0MsA or $\overline{\text{M}}$sA when activated by a command (negative voltage signal) from control point M3 or M5 of the multiplication sequence. Bits R_{41}, R_{42} and R^*_{42} which hold the next base 4 multiplier digit and the mode bit form the inputs to the μMsA logic. Bits Q_{41}, Q_{42} and Q^*_{42} are inputs to the μMsS logic. Its outputs set the MsS selector to either 2MsS, MsS, 0MsS or $\overline{\text{M}}$sS when activated by a command from control point M2, M4 or M6 of the multiplication sequence. The μ decoder is located on chassis Q6C. The θ decoder uses the outputs of the $(\text{H})_1$ and $(\text{H})_2$ status memory elements as its inputs. When its outputs are activated by the add (A) or clear add (B) sequence, the θMsA logic sets the MsA selector to MsA or $\overline{\text{M}}$sA while the θMsS logic sets the MsS selector to 2MsS, $\overline{\text{M}}$sS or $\overline{\text{2M}}$sS. Chassis Q8C contains the θ decoder.

The "Division Predictor and α Logic" (DPα) D-1507 contains the logic and memory elements associated with the mechanism for predicting and recoding

the quotient digits and the so-called α-logic. Utilization of the predictor logic is restricted to the division (D) sequence. The ρMsA logic uses the outputs of the S-adder (σ_{-2}, σ^*_{-2}, σ_{-1}, σ_0, σ^*_0, σ_1 and σ_2) as inputs. Its outputs set the MsA selector to MsA, 0MsA or $\overline{\text{M}}$sA when activated by a command from D7 or D10 of the divide sequence. This same command causes the quotient digit which was predicted on the previous step (-2, 0, 2) to be transferred from the memory element G'_1 and H'_1 to the memory elements G_1 and H_1. It also gates the quotient digit predicted on this step (-1, 0, 1) into G_2 and H_2. The outputs of G_1, H_1, G_2 and H_2 are fed to the quotient digit recoder as shown. Its outputs are then transferred to R^*_{42}, R_{43}, and R_{44}. The ρMsS logic uses the outputs of the A-adder (α_{-1}, α_0, α^*_0, α_2, α^*_2, and α_3) as inputs. When activated by a command from D9 or D11 of the divide sequence, its outputs set the MsS selector to 2MsS, 0MsS or $\overline{2\text{M}}$sS. This command also gates the quotient digit (-2, 0, 2) predicted during this step into G'_1 and H'_1. The α-logic has αov, αnz, nα and nα' as outputs. These outputs are unreliable unless the MsA selector is set to KgA and sufficient time has elapsed for the A-adder outputs to settle. These outputs indicate whether the assimilated value of A is overflowed, non-zero, normalized or almost normalized. The DPα logic is located on chassis A8F, A7F, S8F, S7F, and A6C.

The "Special End Connection Logic" (SEC) D-1527 shows various blocks of logic associated with the shift (F), floating addition (A), and division (D) sequences. The A and S normalization logic is used by the shift (F) sequence to determine overflow during left shifts. The M and R normalization logic is used by the division (D) sequence exclusively. The M normalization logic is used to determine whether the divisor is normalized at the start of a divide order. The R normalization logic is used to determine when the generated quotient appears normalized. The sign comparison logic is used only by the division (D) sequence. Its purpose is to compare the true sign of the partial remainder

in A with the true sign of the divisor in M. The result of this comparison may influence the quotient digit that is inserted in R^{*}_{42}, R_{43}, and R_{44}. The Q and R half-subtractors are used only in case 3 of floating addition. The carry-borrow logic is used in case 4 of floating add (A) and for quotient round-off in the division (D) sequence. Each block of the SEC logic is physically located on the chassis indicated by the drawing.

The "Zero Detect and Round-Off Logic" (ZDR) D-1505 shows the logic used to detect the presence of all zeros in Q, R or S. It also shows the logic used to obtain an unbiased round-off of A. The output of the round-off logic, ρ, is always zero unless the round-off status memory element (RO) is set in the true state, in which case ρ may be 0 or 1. Note that ρ is fed to the low end of the carry generator. The zero detect logic is distributed throughout the Q and S levels of the MAU. The round-off logic is located on the Q7F chassis.

3.1.2 The MAU Bit Path Logic

The bit path logic for the MAU is displayed on six drawings. The standard bit paths are shown on SAS (D-1707) and SQR (D-1706). The non-standard high and low A, S and M bit path logic is shown on HAS (D-1522) and LAS (D-1523) respectively. Likewise, the non-standard high and low Q and R bit path logic is shown on HQR (D-1525) and LQR (D-1524) respectively.

The description of this logic will be register- and selector-oriented. Equations for the most significant, standard, and least significant bit paths are given for each of the five registers and six selectors of the MAU. The connections with the IN(F1) and OUT(FO) registers of Flow Gate Memory are also included.

3.1.2.1 The Fl to M Path

As shown at the top of HAS, SAS, and LAS drawings, the Fl_i input to M_i is the i^{th} output of the Fl flow gate register. This output reflects the contents of the Fl_i memory element at all times. It is transferred to M_i whenever the FlgMEM gate is on, i.e., whenever FlgMEM = 1. On the top of HAS, note that Fl_0 is stored in both M_{-1} and M_0, thus duplicating the sign bit of the incoming fraction.

3.1.2.2 The R to M Path

The logic associated with this path is shown on the top of HQR (D-1525), SQR (D-1706), and LQR (D-1524). It is used to OR or AND the contents of R to M, leaving the result in M. If both the OR and AND are performed, R is simply transferred to M.

$R \vee M$ can be done by gating r_i to M_i if $r_i = 1$, but leaving M_i unchanged if $r_i = 0$. $R \cdot M$ can be performed by gating r_i to M_i if $r_i = 0$, but leaving M_i unchanged if $r_i = 1$. The Boolean expression for gM_i is given below.

$$gM_i = r_i\ (R \vee MgM) \vee \bar{r}_i\ (R \cdot MgM) \qquad -1 \leq i \leq 44$$

It is clear from this expression that $R \vee M$ is performed when $R \vee MgM = 1$, and $R \cdot M$ is performed when $R \cdot MgM = 1$. If $R \vee MgM = R \cdot MgM = RgM = 1$, the contents of R are transferred to M.

3.1.2.3 The M Register

The M register may be loaded as described in the two preceding sections. Its outputs feed the MsA and MsS selector logic as discussed in the two sections which follow. Its most significant outputs, m_{-1} and m_0, feed the carry-borrow logic described in section 3.1.13. The m_0 signal is also used

in the conditional logic of A12 in the add control sequence as shown on D-1269. The m_0, m_1 and m_2 signals are inputs to the M normalization logic described in section 3.1.10. The least significant bit, m_{44}, is gated into the most significant F-element of the EM register, EM_7, during E1 step of the exponent arithmetic sequence. This is discussed in sections 3.2 and 5.6.

3.1.2.4 The MsA Selector

The MsA selector logic appears at the top of HAS, SAS, and LAS. It accepts the true and complement outputs of the M register as its inputs. Its odd outputs, b_{i-1} (i even), feed difference amplifiers, (L/$\bar{L}$), whose outputs feed the A adder. Its even outputs, b'_i, feed the fifth level of the carry generator as described in section 3.1.8. The outputs of the carry generator, b_i (i even), feed through difference amplifiers to the A adder. The inputs to the A adder are shown at the bottom of HAS, SAS, and LAS.

Let $i = 2j$ and $0 \leq j \leq 22$. The odd and even outputs of the MsA selector are then defined as follows:

$$b_{i-1} = m_i \ (2MsA) \vee m_{i-1} \ (MsA) \vee \bar{m}_{i-1} \ (\bar{M}sA)$$

$$b'_i = m_{i+1} \ (2MsA) \vee m_i \ (MsA) \vee \bar{m}_i \ (\bar{M}sA)$$

$$b_i = b'_i \vee k_i \ (KgA)$$

When $j = 22$, $m_{i+1} = m_{45} = 0$ so the expression for b'_{44} is modified accordingly.

The k_i signal is one of the 23 outputs of the carry generator added to the contents of A to assimilate the stored carries and round-off bit, ρ, when KgA = 1. The assimilated value of A or A rounded appears as the output, α, of the A adder when the MsA selector mechanism is set to KgA. Since the selector mechanism can only be set to one option at any time, KgA = 1 implies

that 2MsA = MsA = $\overline{\text{MsA}}$ = 0, and hence $b'_i = 0$. When the mechanism is set to 0MsA, none of the options are active and $b_{2j-1} = b_{2j} = 0$ for $0 \leq j \leq 22$.

3.1.2.5 The MsS Selector

The MsS selector logic is shown at the top of HAS, SAS and LAS. Its inputs are the true and complement outputs of the M register. Its outputs, t_i, feed the S adder through difference amplifiers shown near the center of the three drawings.

The following expression for t_i is uniform for all bit positions except the 44th:

$$t_i = m_{i+1}\ (2\text{MsS}) \vee m_i\ (\text{MsS}) \vee \overline{m}_i\ (\overline{\text{MsS}}) \vee \overline{m}_{i+1}\ (\overline{2\text{MsS}})$$

The expression for t_{44} is obtained by noting that $m_{45} = 0$ so $\overline{m}_{45} = 1$.

The MsS selector has the largest number of options and is therefore somewhat more complicated than the other selector mechanisms. (See section 4.2.) Suffice it to say at this point that to achieve $t_i = 0$ for all i, Delayed Control must request either 0MsS or KMsS, depending on the previous setting of the MsS mechanism. If 2MsS, MsS, $\overline{\text{MsS}}$, or $\overline{2\text{MsS}}$ was the previous setting, 0MsS is a valid clear request from DC. This is true even though $\overline{\text{CS}}$ may have been previously requested with $\overline{\text{MsS}}$ to accomplish the NOT instruction. If M$\overline{\text{MsS}}$ was the previous setting, meaning that MsS = $\overline{\text{MsS}}$ = 1, and CS = 0, then DC must request KMsS to obtain $t_i = 0$ for all i.

3.1.2.6 The sA Selector

The inputs to the sA selector logic appear at the bottom of the HAS, SAS, and LAS drawings, and are generally the outputs, σ_i and σ_i^*, of the S adder.

However, variations occur at the high and low ends of the A register as shown on HAS and LAS respectively.

$$A_{2j-1} = \sigma_{2j-3}\,(1/4SsA) \vee \sigma_{2j-1}\,(SsA) \vee \sigma_{2j+1}\,(4SsA)$$

$$A_{2j} = \sigma_{2j-2}\,(1/4SsA) \vee \sigma_{2j}\,(SsA) \vee \sigma_{2j+2}\,(4SsA)$$

$$A^{*}_{2j} = \sigma^{*}_{2j-2}\,(1/4SsA) \vee \sigma^{*}_{2j}\,(SsA) \vee \sigma^{*}_{2j+2}\,(4SsA)$$

$$0 \leq j \leq 21$$

$$A_{43} = \sigma_{41}\,(1/4SsA) \vee \sigma_{43}\,(SsA) \vee \rho_{-1}\,(dl)\,(4SsA)$$

$$A_{44} = \sigma_{42}\,(1/4SsA) \vee \sigma_{44}\,(SsA) \vee \rho_{0}\,(dl)\,(4SsA)$$

$$A^{*}_{44} = \sigma^{*}_{42}\,(1/4SsA) \vee \sigma^{*}_{44}\,(SsA)$$

If OSsA is set, 1/4SsA = SsA = 4SsA = 0 as usual. It must be pointed out that DC does not request an sA option alone. Instead, it requests an sAQ option as described in section 4.2. The sAQ selector mechanism interprets this combined request and activates the appropriate sA and sQ options.

Except for σ_{-3} and σ_{-2}, the σ variables in the above expressions are the outputs of the S adder as discussed in sections 3.1.3, 3.1.4, and 3.1.5. The ρ_{-1} and ρ_0 signals are outputs of the R one-digit assimilator as discussed in section 3.1.6. If R^{*}_{0} contains a zero, $r^{*}_{0} = 0$, $\rho_{-1} \equiv r_{-1}$ and $\rho_0 \equiv r_0$. The dl signal has unit value when the DL status memory element is true. When DL is true (i.e., set to "1"), a double length (meaning A,Q or S,R) left shift is possible. When $\overline{DL}$ is true, dl = 0 and only single length (meaning A or S) left shifts are possible. It is clear from the above equations that A_{43} and A_{44} are set to zero when a left shift is performed with dl = 0.

The logic used to form σ_{-3} and σ_{-2} is shown at the left edge of the HAS drawing.

$$\sigma_{-3} = (s_{-1} \vee t_{-1})\ (\overline{cr})\ (\overline{\partial}) \vee \sigma'_{-3}\ (cr) \vee s\sigma_{-3}\ (\partial)$$

$$\sigma_{-2} = (s_{-1} \vee t_{-1})\ (\overline{cr})\ (\overline{\partial}) \vee \sigma'_{-2}\ (cr) \vee s\sigma_{-2}\ (\partial)$$

The cr and ∂ signals are the true outputs of the CR and ∇ status memory elements respectively. The σ'_{-3} and σ'_{-2} signals are the outputs of the R half-subtractor as described in section 3.1.12. The $s\sigma_{-3}$ and $s\sigma_{-2}$ signals are the most significant outputs of the high S special adder as discussed in section 3.1.4.

We now analyze the bit configuration at the high end of S to establish a basis for some of the associated Boolean expressions. Since the S register includes S_{-3}, S_{-2}, and S^*_{-2}, it may contain numbers

$$s = -8s_{-3} + \sum_{i=-2}^{44} s_i 2^{-i} + \sum_{k=-1}^{22} s^*_{2k} 2^{-2k}$$

in the range $-8 \leq s < \frac{32}{3}$ where the binary point lies between s_0 and s_1 as usual. We note

$$s = -8s_{-3} + 4s_{-2} + 4s^*_{-2} + 2s_{-1} + s'$$

where

$$s' = \sum_{i=0}^{44} s_i 2^{-i} + \sum_{k=0}^{22} s^*_{2k} 2^{-2k}$$

If s^*_{-2} is assimilated over s_{-3} and s_{-2}, we find

$$s = -8(s_{-3} \oplus s_{-2}s^*_{-2}) + 4(s_{-2} \oplus s^*_{-2}) + 2s_{-1} + s'$$

provided $-8 \leq s < \frac{32}{3}$. If we now require

$$(s_{-3} \oplus s_{-2}s^{*}_{-2}) = (s^{-2} \oplus s^{*}_{-2}) = s_{-1} \qquad (A)$$

then $-2 \leq s < \frac{8}{3}$. If instead we require

$$(s_{-3} \oplus s_{-2}s^{*}_{-2}) = (s_{-2} \oplus s^{*}_{-2}) \neq s_{-1} \qquad (B)$$

then $-4 \leq s < -\frac{4}{3}$ or $2 \leq s < \frac{14}{3}$. In the following table, we let A = 1 and B = 1 imply the truth of the statements A and B as given above. Note that the adder relation, $s^{*}_{-1}s_{-1} \equiv 0$, rules out four cases.

s_{-3}	s_{-2}	s^{*}_{-2}	s_{-1}	A	B
0	0	0	0	1	0
0	0	0	1	0	1
0	0	1	0	0	0
- ϴ - -	ϴ - -	± - -	± -	- - -	- - -
0	1	0	0	0	0
0	1	0	1	0	0
0	1	1	0	0	0
- ϴ - -	± - -	± - -	± -	- - -	- - -
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	0	1
- ± - -	ϴ - -	± - -	± -	- - -	- - -
1	1	0	0	0	1
1	1	0	1	1	0
1	1	1	0	1	0
- ± - -	± - -	± - -	± -	- - -	- - -

From this we may conclude the following:

1. If $-\frac{4}{3} < s < 2$, A is true.
2. If $-2 \leq s < -\frac{4}{3}$ or $2 \leq s < \frac{8}{3}$, either A or B is true.
3. If $-4 \leq s < -2$ or $\frac{8}{3} < s < \frac{14}{3}$, B is true.

Furthermore, s_{-1} represents all assimilated bits to its left when A is true, but not when B is true. Thus, s_{-1} always represents all assimilated bits to its left when $-\frac{4}{3} < s < 2$.

If we assume s_{-1} represents the assimilated bits to its left, a right shift into A or A,Q should place $\sigma_{-3} = \sigma_{-2} = (s_1 \vee t_{-1})$ in A_{-1} and A_0 as justified in the next paragraph. A careful examination of the DC Flow Chart (D-1128) will reveal this assumption must be valid during the following control steps: A10 for cases 2 and 4 of floating point addition, F2 for both logical and arithmetic right shifts, and the M3 and M5 steps of the multiply sequence. In all cases, the A-adder output, α, is shifted right into S with S^*_{-2} set to zero prior to the step which right shifts the S-adder output, σ, into A. Since $-1 \leq a < 1$ is true initially, it follows that $-3 \leq \alpha = a \pm 2m < 3$ and $-\frac{3}{4} \leq s = \frac{\alpha}{4} < \frac{3}{4}$. The latter range is well within $-\frac{4}{3} < s < 2$, so s_{-1} must represent all assimilated bits to its left.

To justify $\sigma_{-3} = \sigma_{-2} = s_{-1} \vee t_{-1}$, we consider the stored carry summation in the -3 and -2 positions. Note in section 3.1.4 that σ^*_{-2} is formed in the same way as the stored carry output of a standard base 4 pseudo adder as described in section 3.1.3.

$$\sigma_{-2} = s_{-2} \oplus t_{-2} \oplus (s^*_{-2} \vee s_{-1}t_{-1})$$

$$x_{-3} = (s_{-2} \oplus t_{-2})(s^*_{-2} \vee s_{-1}t_{-1}) \vee s_{-2}t_{-2}$$

$$\sigma_{-3} = s_{-3} \oplus t_{-3} + x_{-3}$$

It is clear that $t_{-3} = t_{-2} = t_{-1}$ and we have shown above that $s_{-3} = s_{-2} = s_{-1}$ with $s^*_{-2} = 0$ in the case of interest. Consequently, we have:

$$\sigma_{-2} = s_{-1} + t_{-1} \oplus s_{-1}t_{-1} = s_{-1} \vee t_{-1}$$

$$x_{-3} = (s_{-1} \oplus t_{-1})(s_{-1}t_{-1}) \vee s_{-1}t_{-1} = s_{-1}t_{-1}$$

$$\sigma_{-3} = s_{-1} \oplus t_{-1} \oplus s_{-1}t_{-1} = s_{-1} \vee t_{-1}$$

In case 3 of floating addition, the augend (subtrahend) must be shifted circularly left until its exponent is decreased to the value of the addend (minuend)

exponent. To accomplish this, CL is set during the initial pass through A9 of the add sequence which permits $\alpha_{-1} \longrightarrow R_{43}$, $\alpha_0 \longrightarrow R_{44}$ and $\sigma_{-1} \longrightarrow Q_{43}$, $\sigma_0 \longrightarrow Q_{44}$. Thus, the most significant bits of the augend (subtrahend) are placed in the least significant bits of Q and R before the addition (subtraction) occurs. When it does occur, the sum (difference) is shifted circularly right until its exponent agrees with that of the augend (subtrahend). A carry or borrow may be propagated to the left during this circular shift. To accomplish this, CR is set during the appropriate pass through A10 causing $\sigma_{-3} = \sigma'_{-3}$ and $\sigma_{-2} = \sigma'_{-2}$ to be placed in A_{-1} and A_0. Likewise, cr = 1 causes $\alpha_{-3} = \alpha'_{-3}$ and $\alpha_{-2} = \alpha'_{-2}$ to be placed in S_{-1} and S_0. The σ'_{-3} and σ'_{-2} signals are outputs of the R half-subtractor, and the α'_{-3} and α'_{-2} signals are outputs of the Q half-subtractor as discussed in section 3.1.12

The ∇ status memory element is set during store clear type instructions (STC, ASC, SSC), during SAL and SEX, and during the correction of the remainder for quotient round-off during the divide sequence. The uses of ∇ during store clear and divide are explained below. The use of ∇ during the execution of SAL or SEX is explained in section 3.1.2.12.

At the beginning of a store clear type instruction, the accumulator contains the result of an addition or subtraction in the case of ASC or SSC and the result of the previous order in the case of STC. In all cases, the accumulator is normalized, A is rounded and stored, and Q is corrected for round-off and placed in A. It is the last step which requires that $\sigma_{-3} = s\sigma_{-3}$ and $\sigma_{-2} = s\sigma_{-2}$. The signals $s\sigma_{-3}$ and $s\sigma_{-2}$ are defined in section 3.1.4.

The round-off bit, ρ, is placed in S_{-3} and S_{-2} during the S1 step which also places Q in S and sets ∇. Since Q is always considered positive, its sign bits, which appear in S_{-3} and S_{-2} at this point, would ordinarily be zero. If $\rho = 1$, a unit must be subtracted from the -2 position of Q to correct for the unit added to the $44^{\underline{th}}$ position of A. This is easily accomplished by making ρ

the sign of Q. When Q is in S, this amounts to making $s_{-3} = s_{-2} = \rho$. During the following right shift into A with the MsS selector set to OMsS, $s_{-3} \longrightarrow A_{-1}$ and $s_{-2} \longrightarrow$ to A_0. OMsS insures that $t_{-1} = 0$ so that $s\sigma_{-3} = s_{-3}$, and $s\sigma_{-2} = s_{-2}$ as discussed in section 3.1.4. Since $\overline{CR}$ and ∇ are true, $\sigma_{-3} = s\sigma_{-3}$ and $\sigma_{-2} = s\sigma_{-2}$ are placed in A_{-1} and A_0 respectively during S10.

At the end of the iterative portion of divide the shifted remainder is placed in S during the final pass through D11. The bits in S_{-3}, S_{-2}, and S^*_{-2} are significant, and must be partially assimilated and shifted right into A during D14. Since OMsS is set in D13, the $s\sigma_{-3}$, $s\sigma_{-2}$, σ^*_{-2}, σ_{-1} and σ_0 outputs of the high S special adder represent a partial assimilation of the bits s_{-3}, s_{-2}, s^*_{-2}, s_{-1}, s_0 and s^*_0. (See section 3.1.4.) Since CR is set to "0" and ∇ is set to "1" in D13, $\sigma_{-3} = s\sigma_{-3}$ and $\sigma_{-2} = s\sigma_{-2}$ are placed in A_{-1} and A_0 respectively during D14.

3.1.2.7 The A Register

With one exception, the inputs to the A register are the outputs, A_i and A^*_i, of the sA selector as shown on the bottom of the HAS, SAS, and LAS drawings. These outputs are gated into the A register whenever gA = 1. The A_{-1} F-element is an exception in that it has two inputs, A_{-1} and a_0. The a_0 signal is gated into A_{-1} whenever $a_0 gA_{-1} = 0$. A zero signal is used since the input and grounded base of the gating difference amplifier have been reversed -- hence the solid dot gate symbol. This gate is used at the end of the (B) and (F) sequences during logical operations to insure that A does not appear overflowed at the entrance to the (K) sequence.

The outputs of the A register feed the A adder and the carry generator. The former is discussed in sections 3.1.3, 3.1.4, and 3.1.5. The latter is described in section 3.1.8.

3.1.2.8 The sS Selector

The inputs to the sS selector appear at the middle of the HAS, SAS, and LAS drawings. In general, they are the outputs, α_i and α^*_i, of the A adder. Variations occur at both the high and low ends as shown on the HAS and LAS drawings.

$$S_{-3} = \rho\ (\mathrm{AsR}) \vee \alpha_{-1}\ (4\mathrm{AsS})$$

$$S_{-2} = \rho\ (\mathrm{AsS}) \vee \alpha_{0}\ (4\mathrm{AsS})$$

$$S^*_{-2} = \alpha^*_{0}\ (4\mathrm{AsS})$$

$$S_{2j-1} = \alpha_{2j-3}\ (1/4\ \mathrm{AsS}) \vee q_{2j-1}\ (\mathrm{QsS}) \vee \alpha_{2j+1}\ (4\mathrm{AsS})$$

Except

$$S_1 = \alpha_{-1}\ (\bar{\lambda})\ (1/4\ \mathrm{AsS}) \vee q_1\ (\mathrm{QsS}) \vee \alpha_3\ (4\mathrm{AsS})$$

$$S_{2j} = \alpha_{2j-2}\ (1/4\ \mathrm{AsS}) \vee q_{2j}\ (\mathrm{QsS}) \vee \alpha_{2j+2}\ (4\mathrm{AsS})$$

$$S^*_{2j} = \alpha^*_{2j-2}\ (1/4\ \mathrm{AsS}) \vee \alpha^*_{2j-2}\ (4\mathrm{AsS})$$

$$0 \leq j \leq 20$$

$$S^*_{42} = \alpha^*_{40}\ (1/4\ \mathrm{AsS}) \vee q^*_{42}\ (\mathrm{QsS}) \vee \alpha^*_{44}\ (4\mathrm{AsS})$$

$$S_{43} = \alpha_{41}\ (1/4\ \mathrm{AsS}) \vee q_{43}\ (\mathrm{QsS}) \vee \nu_{-1}\ (\mathrm{dl})\ (4\mathrm{AsS})$$

$$S_{44} = \alpha_{42}\ (1/4\ \mathrm{AsS}) \vee q_{44}\ (\mathrm{QsS}) \vee \nu_{0}\ (\mathrm{dl})\ 4\mathrm{AsS})$$

$$S^*_{44} = \alpha^*_{42}\ (1/4\ \mathrm{AsS})$$

If OAsS is set, 1/4 AsS = QsS = 4AsS = 0. As in the case of the sA selector, DC does not request an sS option alone. An sSR option is requested

and the sSR selector mechanism activates the appropriate sS and sR selector options as described in section 4.2.

In the above equations, the α signals are the outputs of the A adder except for α_{-3} and α_{-2} which are discussed below. The round-off, ρ, is used to set the S_{-3} and S_{-2} F-elements whenever AsR = 1. This is of no consequence except during store clear type instructions as explained previously in section 3.1.2.6. The q signals are the true outputs of the Q register including q^*_{42}. The latter must be sent to S^*_{42} during the D17 step of the divide sequence in order to assimilate a stored carry that may be held in the quotient when a negative quotient digit is anticipated but fails to arise before the division terminates. (See sections 3.1.17 and 5.13.) The ν_{-1} and ν_0 signals are the outputs of the Q one digit assimilator as described in section 3.1.6. The dl signal is the true output of the "double length" status memory element, DL, which must be set to "1" to permit double length shift operations in the accumulator.

The α_{-3} and α_{-2} signals are formed by the logic shown in the lower left hand corner of the HAS drawing.

$$\alpha_{-3} = (s\alpha)(\overline{cr})\bar{\lambda} \vee \alpha'_{-3}(cr)$$

$$\alpha_{-2} = (s\alpha)(\overline{cr})\bar{\lambda} \vee \alpha'_{-2}(cr)$$

$$s\alpha = b_{-1} \vee a_{-1}\bar{k}_{-1} \vee \bar{a}_{-1}k_{-1}(\overline{SsA})$$

The cr and λ signals are the true outputs of the "circular right," CR, and "logical," Λ, status memory elements. The α'_{-3} and α'_{-2} signals come from the Q half-subtractor as described in section 3.1.12. The $k_{-1} = a_0K_0(KgA)$ signal is in section 3.1.8. The $\overline{SsA}$ signal is obtained as shown on the HAS drawing. $\overline{SsA} = 0$ whenever the sA selector is set so that gA = 1 causes $\sigma_i \longrightarrow A_i$.

The original expression for $s\alpha$ was incorrect. The error was detected

by the Snyder program as described in File No. 481 by J. N. Snyder, D. B. Gillies, R. D. Hill, and P. G. Kruger. The logic shown above is correct and is currently in use.

When $\overline{CR}$, $\overline{\Lambda}$, and 1/4AsS are true, sα must represent all bits to the left of α_{-1} with or without KgA = 1. (KgA = 1 means that the MsA selector is set so that the output of the carry generator is added to A causing the A adder output, α, to represent the assimilated value of A.) In this sense, sα plays the role of $(s_{-1} \vee t_{-1})$ as discussed in section 3.1.2.6. Its function is complicated by the fact that A may be assimilated during the right shift into S. In any case, S_{-1} and S_0 must represent all assimilated bits to the left of S_{-1} after the right shift is complete. S_{-3} and S_{-2} are always set to zero in this case.

The sα signal is gated into the S_{-1} and S_0 F-elements during the following control steps: A9 in cases 2 and 4 of floating addition; F1 during arithmetic right shifts that arise in the execution of STF, SIF, SEQ, or SIA; K1 if A is overflowed in the twos complement sense or if it appears to be zero; M4 and M6; and D6. KgA is set during F1, K1, and D6. Generally, KgA is set during A9 unless d = -1 in which case the θMsA decoder controls the setting of the MsA selector as discussed in section 3.1.14. The μMsA decoder controls the setting of the MsA selector during M4 and M6 as discussed in section 3.1.15, so KgA = 0.

When KgA = 0, sα is logically equivalent to $(s_{-1} \vee t_{-1})$. We shall consider this case first before attempting to explain the more difficult situation when KgA = 1.

During the first pass through A9, we are guaranteed that $-1 \leq a \leq 1$ in case 2 or 4 of floating point addition. The initial steps of the multiplication sequence guarantee that $-1/2 \leq a \leq 1/4$ during M4. Furthermore, the stored carries of A are guaranteed to be zero during M4. During M6, the worst case limits on the contents of A are: $-2/3 < a < 1/3$. Using the results of section 3.1.2.6, we know that a_{-1} represents all assimilated bits to its left when

$-4/3 < a < 2$, and hence during A9, M4 and M6.

We can now show that $s\alpha = a_{-1} \vee b_{-1}$. Since α^*_{-2} is formed in the same manner as any other stored carry output, the α_{-3} and α_{-2} signals should represent a standard base 4 pseudo addition over the bits to the left of a_{-1}. (See sections 3.1.3 and 3.1.4.)

$$\alpha_{-2} = a_{-2} \oplus b_{-2} \oplus (a^*_{-2} \vee a_{-1}b_{-1})$$

$$c_{-3} = (a_{-2} \oplus b_{-2})(a^*_{-2} \vee a_{-1}b_{-1}) \vee a_{-2}b_{-2}$$

$$\alpha_{-3} = a_{-3} \oplus b_{-3} \oplus c_{-3}$$

Since $a_{-3} = a_{-2} = a_{-1}$ and $b_{-3} = b_{-2} = b_{-1}$, the above equations reduce to $\alpha_{-3} = \alpha_{-2} = a_{-1} \vee b_{-1}$. In both case 2 and 4 of floating point addition, the shift after the addition, $a \pm m$, is to the right into S during A9 followed by a left shift into A with nothing added during A10. Thus, even though we cannot always assume $b_{-3} = b_{-2} = b_{-1}$ in case 4 addition, the bits inserted in S_{-1} and S_0 during the A9 step are lost during A10.

This was not a problem with the S to A path as discussed in section 3.1.2.6. The shift following a case 2 or 4 addition from S is always straight into A. A discussion of the peculiarities of case 4 addition is given in sections 3.1.13 and 5.5.

Examination of the DC Flow Chart reveals that KgA = 1 during F1, K1, and D6. The correct overflow and detect zero sequence (K) insures that $-1 \leq a < 1$ during F1 and D6. Therefore, a_{-1} represents all assimilated bits to its left. The true sign of A is $a_{-1} \oplus k_{-1}$ where k_{-1} represents the carry into the -1 position as defined above and in section 3.1.8. Since KgA = 1, $b_{2j+1} = 0$ and $\alpha^*_{2j} = 0$ for $-1 \leq j \leq 22$. Furthermore, $\alpha_{-1} = a_{-1} \oplus k_{-1}$. It follows that $\alpha_{-3} = \alpha_{-2} = \alpha_{-1} = a_{-1} \oplus k_{-1} = s\alpha$ during F1 and D6.

A little more care must be exercised in determining the expression for $s\alpha$ during K1. Since KgA = 1, the assimilated contents of A are shifted right into S. The $s\alpha$ signal is used to set S_{-1} and S_0, and should therefore represent the true sign of A. So long as $-2 \leq a < 2$, it is clear that $\alpha_{-1} = a_{-1} \oplus k_{-1}$ is the true sign of A and $s\alpha = a_{-1} \oplus k_{-1}$ is perfectly valid. However, if a = 2, $\alpha_{-1} = 1$ but $s\alpha$ should be zero.

The (A), (B), (D), (F), (L), (M), and (S) sequences may terminate with an entry to the (K) sequence. The K1 step may or may not include gating $s\alpha$ into S_{-1} and S_0. We are guaranteed the following about the value in the accumulator at the entrance to K1: $-1 \leq a < 1$ for (F) and (L); $-1 \leq a \leq 1$ for (D), (M), and (S); $-2 \leq a < 2$ for A; and $-2 \leq a \leq 2$ for B. Therefore, $s\alpha = a_{-1} + k_{-1}$ is valid in all cases except when the CST order is executed with -1 as the operand.

It is much too expensive to detect the case a = 2. Consequently, we trade precise knowledge of the contents of A for transistors and arrive at a more subtle solution. The following observation is useful in this regard. If the sA selector is set to SsA at the entrance to K1, then except for the entry from (B) it is guaranteed that $-5/4 \leq a \leq 5/4$. This guarantee also applies at the entrance to F1 and D6.

Since the above range is well within $-4/3 < a < 2$, a_{-1} is representative of all assimilated bits to its left. Hence, if $k_{-1} = 1$, a_{-1} and bits to its left must have unit value, i.e., $\bar{a}_{-1}k_{-1} \equiv 0$ when $-5/4 \leq a \leq 5/4$. The expression for $s\alpha = a_{-1} \oplus k_{-1}$ can be simplified to $s\alpha = a_{-1}\bar{k}_{-1}$ in this case.

We now show that $s\alpha = a_{-1}\bar{k}_{-1}$ represents the sign of A when K1 is entered from the (B) sequence. If a binary left shift is executed, BLS, $\bar{k}_{-1} = 1$ while $a_0 gA_{-1}$ insures that $a_{-1} = a_0$, so $s\alpha = a_{-1}$, which is clearly the sign of A. Except for CSB and CST, the other instructions which are executed via the (B) sequence leave A assimilated, i.e., without stored carries, so $\bar{k}_{-1} = 1$ and a_{-1}

represents the sign of A. Both CSB and CST may leave A with a stored carry in A^*_{42} at the exit from the (B) sequence. All other stored carries are zero, so the bit pattern at the high end of A is easily predicted. In fact, a_{-1} represents the sign of A in all cases. Note that $\bar{k}_{-1} = 0$ only when CST is executed with -1 as the operand. Since the result is $a = 2$, $s\alpha = a_{-1}\bar{k}_{-1} = 0$ accurately reflects the sign of A.

In case 2 of floating add the range of the result is $-2 \leq a < 2$ if the exponents of the augend (subtrahend) and addend (minuend) are equal, i.e., $d = -1$. Note that when d is odd, the sA selector is set to 1/4 SsA instead of SsA. The sign of A is determined by $s\alpha = a_{-1} \oplus k_{-1}$. The two examples shown below illustrate how the sign of A may be + if $a_{-1} = 1$ or - if $a_{-1} = 0$.

Initial A ± M	0 1 0 1 1.0 1 0 1 . . . 0 0.0 1 1 1 . . .	0 0 1 1 0.1 1 1 1 . . . 1 1.0 0 0 0 . . .
Right Shift into S with KgA = 0	0 0 1 x 1 1.1 1 1 1 0 0 . . .	0 0 1 x 1 1.0 1 1 1 0 0 . . .
Left Shift into A with KgA = 1	1 0 1 1.0 0 0	1 0 0 1.0 0 0 . . . .
A assimilated at entrance to K1	0 0.0 0 0	1 0.0 0 0 . . . .

The resulting expression for $s\alpha$ is the one given at the beginning of this section. When KgA = 0, $k_{-1} = 0$ and $s\alpha = a_{-1} \vee b_{-1}$. When KgA = 1, $b_{-1} = 0$ and $s\alpha = a_{-1}\bar{k}_{-1}$. When KgA = 1 and SsA = 0, $b_{-1} = 0$ and $s\alpha = a_{-1} \oplus k_{-1}$. Therefore, the value of $s\alpha$ is correctly determined in all cases.

We now examine the other variables in the equations which define the sS selector.

When a case 3 floating addition (subtraction) is performed, CR must

be set before the result can be circularly right shifted in A,Q. As explained in section 3.1.12, α'_{-3} and α'_{-2} are the outputs of the Q half-subtractor and are placed in S_{-1} and S_0 during A9. The equations which define α_{-3} and α_{-2} show that they must equal α'_{-3} and α'_{-2} respectively when cr = 1.

The LRS and SRS orders are logical shift orders which treat the assimilated value of A as a 45 bit data word (α_0, α_1 --- α_{44}) without sign. When a right shift is performed, zeros are inserted at the left ends of A and S. During LRS or SRS, λ = 1 and cr = 0, so $\alpha_{-3} = \alpha_{-2} = 0$. Since KgA = 1 during any pass through F1 including the first, $\alpha^*_{-2} = 0$. The sS equations show that $S_1 = 0$ if 1/4AsS = 1 and λ = 1. It follows that S_{-3}, S_{-2}, S^*_{-2}, S_{-1}, S_0, S^*_0 and S_1 are all set to zero during the first pass through F1 if LRS or SRS is executed with 1/4AsS = 1. Since $\overline{CR}$, $\bar{\nabla}$, and OMsS are true during LRS or SRS, the equations in section 3.1.2.6 show that $\sigma_{-3} = \sigma_{-2} = 0$, so A_{-1} and A_0 are set to zero during F2 if 1/4SsA = 1.

3.1.2.9 The S Register

In all cases the S register F-elements are set according to the outputs of the sS selector when gS = 1. The outputs of the S register feed the S adder (sections 3.1.3, 3.1.4, and 3.1.5); The S zero detect logic (section 3.1.7); and the S normalization logic (section 3.1.10).

3.1.2.10 The sQ Selector

The sQ selector logic appears at the bottom of the HQR, SQR, and LQR drawings. Except for variations at the most and least significant ends, the inputs to this logic are the true outputs of the R register.

$$Q_{-1} = \sigma_{43}(1/4RsQ) \vee \rho_{-1}(RsQ) \vee r_1(4RsQ)$$

$$Q_0 = \sigma_{44}(1/4RsQ) \vee \rho_0(RsQ) \vee r_2(4RsQ)$$

$$Q_0^* = \sigma_{44}^*(1/4RsQ)$$

$$Q_1 = \rho_{-1}(1/4RsQ) \vee r_1(RsQ) \vee r_3(4RsQ)$$

$$Q_2 = \rho_0(1/4RsQ) \vee r_2(RsQ) \vee r_4(4RsQ)$$

$$Q_i = r_{i-2}(1/4RsQ) \vee r_i(RsQ) \vee r_{i+2}(4RsQ)$$

$$3 \leq i \leq 38$$

$$Q_{39} = r_{37}(1/4RsQ) \vee r_{39}(RsQ) \vee d_{41}(4RsQ)$$

$$Q_{40} = r_{38}(1/4RsQ) \vee r_{40}(RsQ) \vee d_{42}(4RsQ)$$

$$Q_{41} = r_{39}(1/4RsQ) \vee d_{41}(RsQ) \vee r_{43}(4RsQ)$$

$$Q_{42} = r_{40}(1/4RsQ) \vee d_{42}(RsQ) \vee r_{44}(4RsQ)$$

$$Q_{43} = d_{41}(1/4RsQ) \vee r_{43}(RsQ) \vee \sigma_{-1}(cl)(4RsQ)$$

$$Q_{44} = d_{42}(1/4RsQ) \vee r_{44}(RsQ) \vee \sigma_0(cl)(4RsQ)$$

If ORsQ is set, $1/4RsQ = RsQ = 4RsQ = 0$. An sQ setting is requested in conjunction with an sA setting as explained in section 4.2.

The σ_{43}, σ_{44} and σ_{44}^* signals are three of four outputs of the low S special adder as described in section 3.1.5. The ρ_{-1} and ρ_0 signals are from the R one digit assimilator as shown at the left center of the HQR drawing. The function of this assimilator is explained in section 3.1.6. The d_{41} and d_{42} signals are from the borrow-subtractor logic as shown in detail on the SEC drawing (D-1527). The function of the borrow-subtractor is discussed in section 3.1.17. When $r_{42}^* = 0$, $d_{41} = r_{41}$ and $d_{42} = r_{42}$. The σ_{-1} and σ_0 signals are two

outputs of the high S special adder as discussed in section 3.1.4. These outputs are gated into Q_{43} and Q_{44} during a circular left shift in case 3 floating addition. It is only under this condition that the "circular left" status memory element, CL, is set to the "1" state, i.e., cl = 1. Otherwise, a left shift into Q places zeros in Q_{43} and Q_{44}.

3.1.2.11 The Q Register

With the exception of Q^*_{42} and Q^*_{44} the outputs of the sQ selector are used to set the F-elements of the Q register when gQ = 1. The $0_g Q_{43} Q_{44}$ signal appearing in the lower right corner of LQR is used to set Q_{43} and Q_{44} to zero when $0_g Q_{43} Q_{44} = 0$; otherwise, it has no effect. $0_g Q_{43} Q_{44} = 1$ is true during all other control steps.

The signal used to set Q^*_{42} when gQ = 1 is given below.

$$Q^*_{42} = r_{42}(\mu r_{41}) \vee r^*_{42}(\mu r_{41}) \vee r^*_{42} \bar{r}_{43} \bar{r}_{44} \delta_2$$

The first two terms of this expression are part of the μMsA decoder logic which is described in section 3.1.15. The multiply status memory element, μ, is set to "1" during the M1 step of the multiply sequence, and is reset to "0" during the final pass through M6. Since $\delta_2 = 0$ except during division, $Q^*_{42} = Q^{*\,\prime}_{42}$ as defined in section 3.1.15 when $\mu = 1$.

The last term in the above expression is associated with the borrow-subtractor logic as described in section 3.1.17. The divide status memory element, Δ_2, is set (i.e., $\delta_2 = 1$) during D10 of the divide control sequence and reset during D14. It serves a dual purpose in division as discussed in section 3.1.17 and 5.13. Since $\mu = 0$ except during multiplication, $Q^*_{42} = Q^{*\,\prime}_{42} = r^*_{42} \bar{r}_{43} \bar{r}_{44}$ when $\delta_2 = 1$.

The Q^*_{44} signal sets the Q^*_{44} F-element when gQ = 1. It is an output of the R half-subtractor logic which is described in section 3.1.12. The Boolean expression for Q^*_{44} is given below for completeness.

$$Q^*_{44} = (cr)(\bar{r}_{43})(\bar{r}_{44})[r^*_{44}(OMsS) \vee \bar{s}_{-1}t_{-1}(\overline{OMsS})]$$

The Q register outputs feed the sS and sR selector logic (sections 3.1.2.8 and 3.1.2.12); the Q zero detector and round-off logic (section 3.1.7); the Q one digit assimilator (section 3.1.6); the μMsS decoder (section 3.1.15); and the Q half-subtractor (section 3.1.12).

3.1.2.12 The sR Selector

The logic of the sR selector is shown in the middle of the HQR, SQR, and LQR drawings. In general, the inputs to this logic are the outputs of the Q register and the A adder. There are variations at the most and least significant ends of the selector as usual.

$$R_{-1} = \alpha_{43}(\bar{\delta})(1/4QsR) \vee \alpha_{-1}(AsR) \vee q_1(4QsR)$$

$$R_0 = \alpha_{44}(\bar{\delta})(1/4QsR) \vee \alpha_0(AsR) \vee q_2(4QsR)$$

$$R^*_0 = \alpha^*_{44}(1/4QsR)$$

$$R_1 = \nu_{-1}(1/4QsR) \vee \alpha_1(AsR) \vee q_3(4QsR)$$

$$R_2 = \nu_0(1/4QsR) \vee \alpha_2(AsR) \vee q_4(4QsR)$$

$$R_i = q_{i-2}(1/4QsR) \vee \alpha_i(AsR) \vee q_{i+2}(4QsR)$$

$$3 \leq i \leq 42$$

$$R_{43} = q_{41}(1/4QsR) \vee \alpha_{43}(AsR) \vee \rho_{43}(\delta_1)(4QsR) \vee \alpha_{-1}(c1)(4QsR)$$

$$R_{44} = q_{42}(1/4QsR) \vee \alpha_{44}(AsR) \vee \rho_4\ (\delta_1)(4QsR) \vee \alpha_0(c1)(4QsR)$$

When 0QsR = 1, 1/4QsR = AsR = 4QsR = 0 as usual. An sR selector option as explained in section 4.2.

The α signals are outputs of the A adder as described in sections 3.1.3, 3.1.4 and 3.1.5. The ν_{-1} and ν_0 signals are outputs of the Q one digit assimilator which is shown in the lower left corner of the HQR drawing and is discussed in section 3.1.6. The ρ_{43} and ρ_{44} signals are outputs of the quotient "bit" recoder as described in section 3.1.17. The δ_1 signal is derived from the division sequence logic. As explained in section 3.1.17, δ_1 = 1 during D9, D10, D11, D12. (See Divide Control Logic D6-D12, D-1272.) Since $\overline{CL}$ and 4QsR are both true during these control steps, ρ_{43} and ρ_{44} set R_{43} and R_{44} when gR = 1. CL is set to "1" prior to the circular left shift in case 3 floating addition to permit α_{-1} and α_0 to set R_{43} and R_{44} as discussed in section 3.1.12. Otherwise, zeros are placed in R_{43} and R_{44} when 4QsR = gR = 1.

$\overline{\delta}$ is the complement output of the ∇ status memory element and is used to set R_{-1} and R_0 to zero during the S2 step of the store sequence. This step is used only during the execution of an SAL (store Q) or SEX (store exponent) order. In either case, Q contains the least significant half of the double length accumulator. During S2 the contents of Q are shifted right into R. Since all bits of Q have positive weight, the fraction q_{-1}, q_0 . . . q_{41}, q_{42} must be stored with a positive sign. As Q is shifted right into R during S2, ∇ is set so $\overline{\delta}$ = 0, causing zeros to be gated into R_{-1} and R_0. Note that $\overline{\nabla}$ is set during S10 and that $\overline{\delta}$ = 1 is always true during all other control steps in which 1/4QsR = 1. The ∇ status memory element serves another purpose during the divide sequence as explained in sections 3.1.2.6 and 5.13.

3 1 2 13 The R Register

Except for R^*_{42} and R^*_{44}, the outputs of the sR selector are used to set the R register F-elements when gR = 1.

$$R^*_{42} = \mu[q_{41}q_{42}(\overline{KgA}) \vee q_{41}q^*_{42}(\overline{KgA}) \vee \alpha_{43}\alpha_{44}(KgA)] \vee \delta_2\beta_{42}$$

The term associated with the μ signal is part of the μMsS decoder. When $\mu = 1$, $\delta_2 = 0$ so that $R^*_{42} = R^{*\prime}_{42}$ as defined in section 3.1.15. When $\delta_2 = 1$, $\mu = 0$ so $R^*_{42} = \beta_{42}$ which is an output of the quotient "bit" recoder as defined in section 3.1.17.

The signal used to set R^*_{44} when gR = 1 is defined below. It is an output of the Q half-subtractor which is discussed in section 3.1.12.

$$R^*_{44} = (cr)(\bar{q}_{43})(\bar{q}_{44})[q^*_{44}(OMsA) \vee \bar{a}_{-1}b_{-1}(\overline{OMsA})]$$

In addition to the sQ selector logic (section 3.1.2.10), the outputs of the R register feed the FO register and the associated store logic which is discussed below; the R to M logic (section 3.1.2.2); the R one digit assimilator (section 3.1.6); the R normalization detector (section 3.1.10); the R zero detect logic (section 3.1.7); the μMsA decoder (section 3.1.15); the borrow-subtractor (section 3.1.17); and the R half-subtractor (section 3.1.12).

3 1 2 14 The R to FO Path

The logic in the path between the R register and the FO (OUT) register is commonly referred to as store logic. It is shown at the top of the HQR, SQR, and LQR drawings The inputs to FO are designated as r_i' When RESgFO = 1, r_i' is stored in FO_i.

$$r_0' = r_{-1}r_0 \vee r_0{}^j$$

$$r_1' = \bar{r}_0 r_1 \vee r_1 \bar{r}_2 \vee r_1 \bar{n}$$

$$r_2' = \bar{r}_{-1} r_0 \bar{j} \vee \bar{r}_0 r_2 \vee \bar{r}_1 r_2 \vee r_2 \bar{n}$$

$$r_i' = r_i \qquad (3 \leq i \leq 38)$$

$$r_i' = \bar{x} r_i \vee x(es_7) \qquad (39 \leq i \leq 44)$$

The n, j, and x signals are the true outputs of the N, J, and X status memory elements respectively. Since RESgFO = 1 only during the S9 step of the store sequence, the settings of N, J, and X at that time will determine the way in which R is transferred to FO. X is never true during S9 except when SEX (store exponent) is executed.

The six store orders which normalize the accumulator before storage takes place are ASC, SSC, STR, XCH, STC, and STN. For these orders N, $\bar{J}$, and $\bar{X}$ are true during the S9 step. Therefore, $r_0' = r_{-1} r_0$, $r_1' = \bar{r}_0 r_1 \vee r_1 \bar{r}_2$, $r_2' = \bar{r}_{-1} r_0 \vee \bar{r}_0 r_2 \vee \bar{r}_1 r_2$, and $r_i' = r_i$ for $3 \leq i \leq 44$ when RESgFO = 1. Normalization insures that the fraction in A,Q satisfies: $-1 \leq f < -1/4$, $f = 0$, or $1/4 \leq f < 1$. $\bar{A}$ and $\bar{J}$ are true for all six orders so A is rounded during S1. Consequently, the fraction which is stored may be in the range: $-1 \leq f \leq -1/4$, $f = 0$, or $1/4 \leq f \leq 1$. If f does not equal $-1/4$ or $+1$, then $r_0' = r_0$, $r_1' = r_1$ and $r_2' = r_2$. If $f = -1/4$, $r_0' = 1$, $r_1' = 0$, $r_2' = 0$, and $r_i' = 0$ for $3 \leq i \leq 44$. In other words, $f = -1/4$ is stored as -1, and the exponent is decreased by one to compensate in steps S5 and S7. If $f = +1$, $r_0' = 0$, $r_1' = 0$, $r_2' = 1$, and $r_i' = 0$ for $3 \leq i \leq 44$. Thus, $f = +1$ is stored as $+1/4$, and the exponent is increased by one to compensate in steps S5 and S7.

The STU, STF, and SEQ orders do not normalize the accumulator but they do round A on the way to R. $\bar{J}$ and $\bar{N}$ are true for STU while J and $\bar{N}$ are true for STF and SEQ.

During S9 for STU, $r_0' = r_{-1}r_0$, $r_1' = r_1$, $r_2' = \bar{r}_{-1}r_0 \vee r_2$, and $r_i' = r_i$ for $3 \leq i \leq 44$. If $-1 \leq f < 1$, $r_i' = r_i$ for $0 \leq i \leq 44$. If $f = +1$, $r_0' = 0$, $r_1' = 0$, $r_2' = 1$, and $r_i' = 0$ for $3 \leq i \leq 44$. Therefore, +1 is stored as +1/4 and the exponent is increased by one to compensate in steps S5 and S7.

During S9 for STF and SEQ, $r_i' = r_i$ for $0 \leq i \leq 44$. Therefore, the rounded fraction in R is stored Modulo 2.

SAM, SIF, and SIA store A or A shifted without normalization or round-off. Since $\bar{J}$ and $\bar{N}$ are true for SAM, the store logic between R and FO behaves as described for the STU order. J and $\bar{N}$ are true for SIF and SIA so the behavior of the store logic is the same as described for the STF and SEQ orders.

The SAL order right shifts the contents of Q by one base 4 position on the way to R and inserts zeros in R_{-1} and R_0. $\bar{J}$ and $\bar{N}$ are true during S9, so R to FO logic behaves as described for the STU order. However, $r_{-1} = r_0 = 0$ in all cases, so it is never necessary to modify the exponent.

The SRM order stores R Modulo 2. J and $\bar{N}$ are true during S9, so the R to FO logic behaves as described for the STF and SEQ orders. This order is primarily used to store the fractional remainder, r, immediately after a divide order. The Modulo 2 store causes no difficulty since $|r| \leq |d/2|$, and the fractional divisor, d, is always less than or equal to unity in magnitude.

The SEX order is used to store an 8 bit exponent in the last 13 bit quarter word of FO with the sign bit, es_7, duplicated in bit positions FO_{39}, FO_{40}, FO_{41}, FO_{42}, FO_{43}, and FO_{44}. The signals es_6 through es_0 are stored in FO_{45} through FO_{51} as usual. The overwrite of r_{39} through r_{44} is accomplished by the logic shown at the top of the LQR drawing and defined at the beginning of this section. X is true during the S9 control step only when SEX is executed. For this order the first three quarter words of FO are generally of no importance. As for SAL, ∇ is set to "1" during S2, so R_{-1} and R_0 contain zeros during S9. Since J and $\bar{N}$ are true, bits r_0 through r_{38} are stored in FO Modulo 2.

3.1.3 Standard Base 4 Pseudo Adders

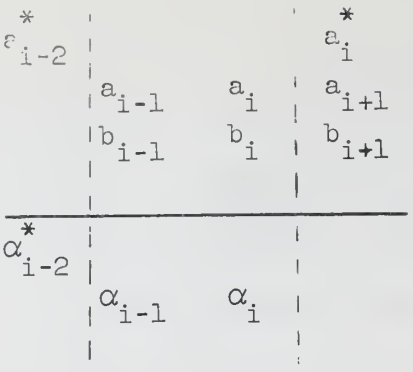
The A and S adders extend over 46 and 48 bit positions respectively. They are composed of a high special adder, a series of 21 standard base 4 pseudo adders, and a low special adder. The A and S inputs to these adders as well as their outputs are in stored carry representation. For a discussion of stored carry representation and its ramifications, the reader is referred to File 263 by J. E. Robertson and G. Metze.

We now consider the base 4 pseudo A-adder as shown on the right half of STA (D-1265). Remember that no attempt is made to distinguish signals which are logically identical but electronically different.

Let a_i represent the true output of A_i and let b_i represent the true output of the MsA selector OR in the $i^{\underline{th}}$ position (i even). Note that b_i depends on the setting of the MsA selector. In the list shown below, m_i represents the true output of M_i.

Selector Setting	Selector Output	Selector Setting	Selector Output
2MsA:	$b_i = m_{i+1}$	$\overline{\text{M}}$sA:	$b_i = \overline{m}_i$
MsA:	$b_i = m_i$	KgA:	$b_i = k_i$ (i even)
0MsA:	$b_i = 0$		$b_i = 0$ (i odd)

The inputs to the standard base 4 pseudo A-adder are a^*_{i-2}, a_{i-1}, a_i, $\overline{a}^*_i$, b_{i-1}, b_i, c_i and $\overline{d}_i$. Since we are examining an A-adder, the carry inhibit input ($\overline{\overline{KI}}$) also appears. The outputs of the base 4 pseudo adder are α^*_{i-2}, α_{i-1}, α_i, c_{i-2}, and $\overline{d}_{i-2}$ where α^*_{i-2} is the stored carry and c_{i-2} represents the carry sent to the adder on the left. The significance of $\overline{d}_{i-2}$ will become apparent below. The addition performed may be visualized as follows:



Let $c_i = a^*_i \vee a_{i+1}b_{i+1}$ and $\bar{d}_i = \bar{a}_{i+1} \vee \bar{b}_{i+1}$. It follows that

$$\alpha_i = a_i \oplus b_i \oplus c_i$$

$$\alpha_i = (a_i\bar{b}_i \vee \bar{a}_i b_i)\overline{a^*_i d_i} \vee (a_i b_i \vee \bar{a}_i\bar{b}_i)c_i$$

The carry into the $(i-1)^{\underline{th}}$ position is $c_{i-1} = a_i b_i \vee c_i(a_i\bar{b}_i \vee \bar{a}_i b_i)$. In terms of this carry and its complement we have

$$\alpha_{i-1} = a_{i-1} \oplus b_{i-1} \oplus c_{i-1}$$

$$\alpha_{i-1} = (a_{i-1}\bar{b}_{i-1} \vee \bar{a}_{i-1}b_{i-1})\bar{c}_{i-1} \vee (a_{i-1}b_{i-1} \vee \bar{a}_{i-1}\bar{b}_{i-1})c_{i-1}$$

The stored carry is formed as

$$\alpha^*_{i-2} = (a_{i-1}\bar{b}_{i-1} \vee \bar{a}_{i-1}b_{i-1})(c_{i-1})(\overline{KI})$$

The carry which is sent to the next adder to the left is

$$c_{i-2} = a^*_{i-2} \vee a_{i-1}b_{i-1}$$

For convenience, the partial complement of c_{i-2} is also formed as

$$\bar{d}_{i-2} = \bar{a}_{i-1} \vee \bar{b}_{i-1}$$

Note that the carry propagation chain is broken by storing the carry

α^*_{i-2} which arises to the right of the $(i-1)^{\underline{th}}$ position. The c_{i-2} carry can only arise as the consequence of a stored carry, a^*_{i-2}, or the condition $a_{i-1} = b_{i-1} = 1$. It is important to note that the condition $a^*_{i-2} = a_{i-1} = 1$ can never arise. This is quickly proven by showing that $\alpha_{i-1}\alpha^*_{i-2} \equiv 0$. This identity is called the adder relation.

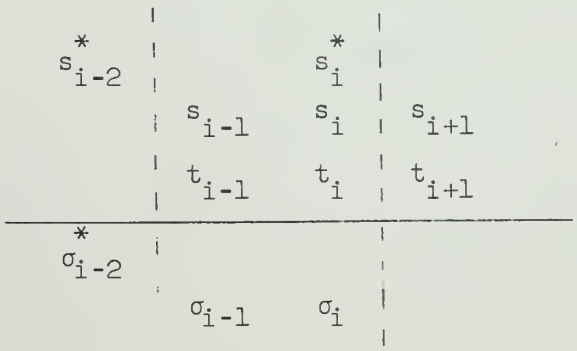
When KgA = 1 (i.e., when the MsA selector is set to KgA such that b_i equals the k_i output of the carry generator if i is even and $b_i = 0$ if i is odd), the carry inhibit signal $\overline{KI} = 0$. It is clear that $\alpha^*_{i-2} = 0$ under these conditions. The necessity of the $\overline{KI}$ input is explained in section 4.2.

A description of the standard S-adder is almost identical. The adder shown on the right half of STS (D-1266) is considered below.

Let s_i be the true output of S_i and let t_i be the true output of the MsS selector OR in the $i^{\underline{th}}$ position. The value of t_i depends on the setting of MsS as shown below.

Selector Setting	Selector Output	Selector Setting	Selector Output
2MsS:	$t_i = m_{i+1}$	$\overline{\text{MsS}}$:	$t_i = \overline{m}_i$
MsS:	$t_i = m_i$	$\overline{\text{2MsS}}$:	$t_i = \overline{m}_{i+1}$
0MsS: } KMsS: }	$t_i = 0$	$\overline{\text{MMsS}}$:	$t_i = 1$
		CS:	$t_i = 0$

The inputs to the standard base 4 pseudo S-adder are s^*_{i-2}, s_{i-1}, s_i, $\overline{s}^*_i$, t_{i-1}, t_i, x_i, $\overline{y}_i$. The outputs are σ^*_{i-2}, σ_{i-1}, σ_i, x_{i-2}, and $\overline{y}_{i-2}$. In this notation, σ^*_{i-2} is the stored carry and x_{i-2} is the carry transmitted to the adder on the left. The significance of $\overline{y}_{i-2}$ is explained below.



Let $x_i = s_i^* \vee s_{i+1}t_{i+1}$ and $\bar{y}_i = \bar{s}_{i+1} \vee \bar{t}_{i+1}$. It follows that

$$\sigma_i = s_i \oplus t_i \oplus x_i$$

$$\sigma_i = (s_i\bar{t}_i \vee \bar{s}_i t_i)\bar{s}_i\bar{y}_i \vee (s_i t_i \vee \bar{s}_i\bar{t}_i)x_i$$

The carry into the $(i-1)^{\underline{th}}$ position is

$$x_{i-1} = s_i t_i \vee x_i(s_i\bar{t}_i \vee \bar{s}_i t_i)$$

Using this carry and its complement we have

$$\sigma_{i-1} = s_{i-1} \oplus t_{i-1} \oplus x_{i-1}$$

$$\sigma_{i-1} = (s_{i-1}\bar{t}_{i-1} \vee \bar{s}_{i-1}t_{i-1})\bar{x}_{i-1} \vee (s_{i-1}t_{i-1} \vee \bar{s}_{i-1}\bar{t}_{i-1})x_{i-1}$$

The stored carry is given by

$$\sigma_{i-2}^* = (s_{i-1}\bar{t}_{i-1} \vee \bar{s}_{i-1}t_{i-1})x_{i-1}$$

The carry which is sent to the adder on the left and its partial complement are formed as

$$x_{i-2} = s_{i-2}^* \vee s_{i-1}t_{i-1}$$

and

$$\bar{y}_{i-2} = \bar{s}_{i-1} \vee \bar{t}_{i-1}$$

These Boolean expressions are identical to those for the A-adder except that σ_{i-2}^* is not a function of $\overline{KI}$.

3.1.4 High Special Adders

The high A special adder is shown on the right half of HAD (D-1526).

Its inputs are a_{-1}, $\bar{a}_{-1}$, a_0, $\bar{a}^*_0$, b_{-1}, $\bar{b}_{-1}$, b_0, $\bar{b}_0$, c_0, $\bar{d}_0$, and $\overline{KI}$. Its outputs are α^*_{-2}, α_{-1}, $\bar{\alpha}_{-1}$, α_0, and $\bar{\alpha}_0$. The Boolean expressions for its outputs are the same as those for the standard A-adder with i = 0. Since there is no adder to the left, c_{-2} and $\bar{d}_{-2}$ are not formed. The amplifiers and cable drivers providing fanout for the α_{-1} and α_0 signals are the only other special features of this adder.

The special high S-adder is also quite similar to the standard S-adder. Its inputs are s_{-3}, s_{-2}, $\bar{s}_{-2}$, s^*_{-2}, $\bar{s}^*_{-2}$, s_{-1}, $\bar{s}_{-1}$, s_0, $\bar{s}_0$, $\bar{s}^*_0$, x_0, $\bar{y}_0$, t_{-1}, $\bar{t}_{-1}$, t_0, $\bar{t}_0$, m_{-1}, $\bar{m}_{-1}$, m_0, and $\bar{m}_0$. Its outputs are $s\sigma_{-3}$, $s\sigma_{-2}$, $\overline{s\sigma}_{-2}$, σ^*_{-2}, $\bar{\sigma}^*_{-2}$, σ_{-1}, $\bar{\sigma}_{-1}$, σ_0, and $\bar{\sigma}_0$. The Boolean expressions for σ^*_{-2}, σ_{-1}, and σ_0 are the same as those for the standard S-adder with i = 0. The complements of these signals are obtained with difference amplifiers as shown on D-1526. The special outputs $s\sigma_{-3}$, $s\sigma_{-2}$, and $\overline{s\sigma}_{-2}$ are used only during division and store orders as discussed in sections 3.1.2.6 and 5.13. None of these outputs were anticipated in the original design.

The need to include $s\sigma_{-2}$ and $\overline{s\sigma}_{-2}$ as inputs to the ρMsA division predictor was pointed out by R. H. Farrell. The logic used to realize these signals was valid for all division orders and operands except for an NDV order with an operand of -1 or a VID order with a rounded accumulator of +1. In either case the fractional division is performed with a divisor M = +1. Using the original logic, this condition gave rise to erroneous quotients whenever the dividend was such as to cause 2MsS or $\overline{2M}$sS followed by MsA or $\bar{M}$sA to be selected during the division process. This error was first discovered by the group working on a slightly modified version of the New Illinois Computer at the Weizmann Institute of Science at Rehovoth, Israel. As reported by M. Melman, an NDV order gave incorrect results when the initial dividend and divisor were -1. This led to the discovery of other cases. The expressions for $s\sigma_{-2}$ and $\overline{s\sigma}_{-2}$ shown below are valid for all division orders and operands.

The $s\sigma_{-3}$ output was created after a division error was detected by J. E. Robertson's ASMD test routine. The trouble in this case was traced to

the fact that the adder relation $a_0^* a_1 \equiv 0$ was occasionally violated during the D14 step of the divide sequence. To avoid this, it is necessary to assimilate over S_{-3}, S_{-2}, and S_{-2}^* during the down right shift into A_{-1}, A_0, and A_0^* whenever the ∇ status memory element is in the "1" state. (See the expressions for the inputs to A_{-1}, A_0, and A_0^* in section 3.1.2.6)

It is most informative to develop the Boolean expressions for these special outputs separately. We shall consider the expressions for $s\sigma_{-2}$ and $\overline{s\sigma}_{-2}$ first.

Since σ_{-2}^* is formed in the standard manner, the theoretical expression for $s\sigma_{-2}$ (i.e., the σ output in the -2 position) is

$$s\sigma_{-2} = s_{-2} \oplus t_{-2} \oplus x_{-2}$$

where

$$x_{-2} = s_{-2}^* \vee s_{-1} t_{-1}$$

The $s\sigma_{-2}$ signal is used only during the D7, D10, and D14 steps of division and the S2 step of store. Except in D7 and D10, the t_i ($i = -3, -2, -1, \ldots$) signals are 0 since 0MsS is selected. In the D7 and D10 steps 2MsS, 0MsS, or $\overline{2M}sS$ may be selected. The first and last of these options require special attention. If $-1 \leq m < +1$, $t_{-2} = t_{-1}$ for 2MsS or $\overline{2M}sS$. If $m = +1$, $t_{-2} \neq t_{-1}$ except when 0MsS is selected. The following expression for t_{-2} is correct for the three selector options of interest when $-1 \leq m \leq +1$.

$$t_{-2} = t_{-1}\overline{(\bar{m}_{-1} m_0)} \vee t_0(\bar{m}_{-1} m_0) = t_{-1} m_{-1} \vee t_{-1}\bar{m}_0 \vee t_0 \bar{m}_{-1} m_0$$

This expression is clearly incorrect if we were concerned with the MsS or $\bar{M}sS$ selector options. In that event it would be necessary to replace t_0 with t_1 in the above expression. Since the MsS and $\bar{M}sS$ options are never used in conjunction with the $s\sigma_{-2}$ signal in the present design, the availability of the

t_0 signal as compared with the t_i signal argues in favor of the above expression. Therefore, $s\sigma_{-2}$ and $\overline{s\sigma}_{-2}$ are formed as

$$s\sigma_{-2} = [s_{-2}\bar{x}_{-2} \vee \bar{s}_{-2}x_{-2}] \oplus t_{-2}$$

$$\overline{s\sigma}_{-2} = [s_{-2}\bar{x}_{-2} \vee \bar{s}_{-2}x_{-2}] \equiv t_{-2}$$

The term in brackets is called cy on the HAD drawing. By using the restoring EXCLUSIVE-OR and EQUIVALENCE circuits we avoid forming t_{-2} and insure the proper voltage level for $s\sigma_{-2}$ and $\overline{s\sigma}_{-2}$ as inputs to the division predictor. The carry signals x_{-2} and $\bar{x}_{-2}$ are formed in the standard manner as shown on STS (D-1266).

We now consider the $s\sigma_{-3}$ signal. This output of the high S special adder is not equivalent to the σ_{i-3} output of a standard S-adder except when OMsS is selected. However, OMsS is the selector option when $s\sigma_{-3}$ is used in steps D14 and S2. The theoretically correct expression for $s\sigma_{-3}$ is

$$s\sigma_{-3} = s_{-3} \oplus t_{-3} \oplus x_{-3}$$

where

$$x_{-3} = s_{-2}t_{-2} \vee x_{-2}(s_{-2} \oplus t_{-2})$$

and

$$x_{-2} = s^{*}_{-2} \vee s_{-1}t_{-1}$$

Since t_i signals are 0 in the cases of interest, this expression can be simplified to

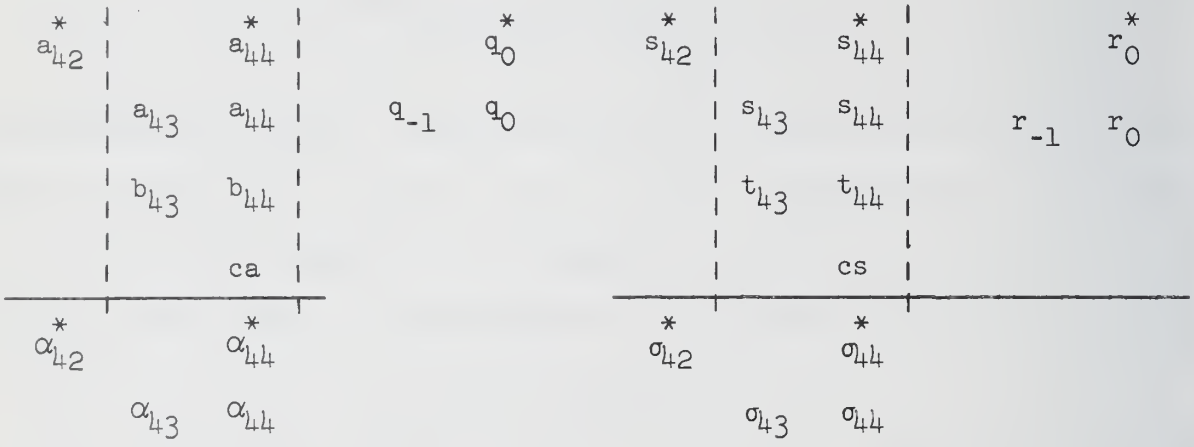
$$s\sigma_{-3} = s_{-3} \oplus ky$$

where

$$ky = s_{-2}s^{*}_{-2}$$

3.1.5 Low Special Adders

These adders were designed for the least significant base 4 digital position of A and S. The following bit configurations must be examined in this position.



The variables q_i and r_i represent the true outputs of Q_i and R_i. Bits (ca) and (cs) are the complement carry bits which must be added in the $44^{\underline{th}}$ position when doing a subtraction of M or 2M from A or S in complement arithmetic. It was noted earlier that ca = 1 when the MsA selector is set to $\overline{M}sA$, and that cs = 1 when the MsS selector is set to $\overline{Ms}S$, $\overline{2M}sS$, or CS.

The low special adders are designed such that the adder relation is never broken within the limits of A and S provided $a^*_{44} = s^*_{44} = 0$ prior to a subtraction and left shift; and such that a carry can never arise from Q or R, i.e., $q_{-1}q_0q^*_0 \equiv r_{-1}r_0r^*_0 \equiv 0$. A subtraction followed by a left shift can only occur in division during the transfer from A to S. The subnormalization step prior to the iterative loop of divide sets $a^*_{44} = 0$ and therefore precludes the possibility of a broken adder relation within the limits of A and S. The adder relation may be broken between a^*_{44} and q_{-1} or s^*_{44} and r_{-1}; however, this is of no concern, since the second restriction prevents a carry from propagating into the $44^{\underline{th}}$ position from Q or R.

The cases of interest are those involving four units in the $44^{\underline{th}}$ position. The outputs of the special adders in each of these cases are shown below.

			(1)	(2)	(3)	(4)			
		a^*_{44}	1	1	1	1			s^*_{44}
	a_{43}	a_{44}	11	01	11	01		s_{43}	s_{44}
	b_{43}	b_{44}	11	01	01	11		t_{43}	t_{44}
		ca	1	1	1	1			cs
α^*_{42}		α^*_{44}	1 0	1 0	1 1	1 1	σ^*_{42}		σ^*_{44}
	α_{43}	α_{44}	00	00	01	01		σ_{43}	σ_{44}

In cases (1) and (2), α^*_{44} and σ^*_{44} equal 0 as they do in all cases where fewer than four units appear in the $44^{\underline{th}}$ column. In case (1) the two units in the $43^{\underline{rd}}$ column generate a carry input to the standard base 4 pseudo adder over bits 40^*, 41, and 42. Therefore, the outputs of the special adder do not reflect the presence of these two units. In cases (3) and (4) the outputs of the special adder must represent a sum of six units without violating the adder relation. This is accomplished by allowing a stored carry to appear in the $44^{\underline{th}}$ position of the adder output. If a right shift follows the subtraction indicated by (3) and (4), the restriction $q_{-1}q_0q^*_0 \equiv 0$ or $r_{-1}r_0r^*_0 \equiv 0$ is clearly satisfied.

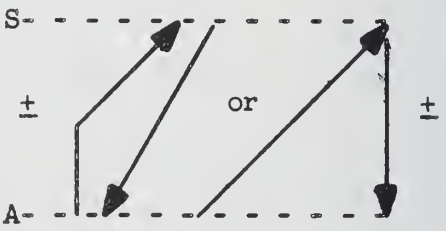
A left shift following either of these subtractions could result in a violated adder relation in S or A. For example, if $q_{-1} = 1$ and $\alpha^*_{44} = 1$, then a left shift into S would produce $\alpha^*_{44} \longrightarrow s^*_{42} = 1$ and $q_{-1} \longrightarrow s_{43} = 1$ such that $s^*_{42}s_{43} \not\equiv 0$. This is why a^*_{44} and s^*_{44} must be 0 at the start of any series of subtractions and left shifts. Since $r_{-1}r_0r^*_0 \equiv q_{-1}q_0q^*_0 \equiv 0$, it is possible to assimilate q^*_0 or r^*_0 during the left shift into S or A such that $0 \longrightarrow s^*_{44}$ or $0 \longrightarrow a^*_{44}$. (See section 3.1.6.) Therefore, the requirement that

$a^*_{44} = s^*_{44} = 0$ at the start of any subtraction and left shift sequence is sufficient to guarantee that the adder relation will not be violated within the limits of A and S.

If the subtractions indicated by (3) and (4) are followed by a straight shift into the next register (which must be from S to A), $a^*_{44}q_{-1}$ may not be identically zero. This could cause trouble if it were followed immediately by a subtraction of type (3) or (4) and a left shift into S. This subtract-shift pattern does not occur in the present realization of Delayed Control. The general add-subtract-shift patterns which do occur are outlined below.

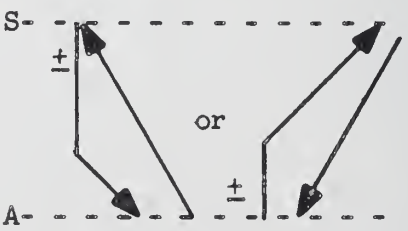
Cases 2 and 4 of Floating Add

(A)



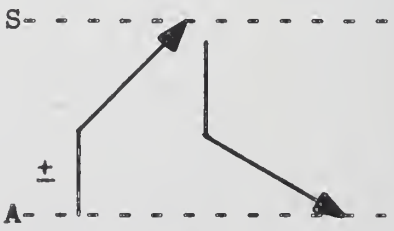
Case 3 of Floating Add

(A)



General Step of Multiply

(M)



Last Step of Multiply

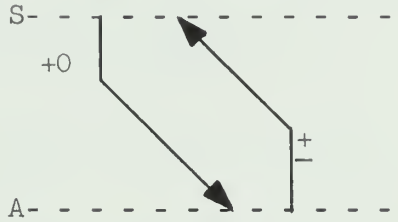
(M)

General Step of Divide

(D)

Remainder Round-Off Connection

(D)



The truth table for the low special adder is shown below. The $c_{43}(x_{43})$ function represents the carry into the $43^{\underline{rd}}$ position. The outputs $\alpha^*_{42}(\sigma^*_{42})$, $\alpha_{43}(\sigma_{43})$, $c_{42}(x_{42})$, and $\bar{d}_{42}(\bar{y}_{42})$ are formed in the standard manner.

a_{43} / s_{43}	b_{43} / t_{43}	a_{44} / s_{44}	b_{44} / t_{44}	a^*_{44} / s^*_{44}	ca / cs	α^*_{44} / σ^*_{44}	α_{44} / σ_{44}	c_{43} / x_{43}	α_{42} / σ_{42}	α^*_{42} / σ^*_{42}
		0	0	0	0	0	0	0		
		0	0	0	1	0	1	0		
		0	0	1	0	0	1	0		
		0	0	1	1	0	0	1		
		0	1	0	0	0	1	0		
		0	1	0	1	0	0	1		
		0	1	1	0	0	0	1		
		0	1	1	1	0	1	1		
		1	0	0	0	0	1	0		
		1	0	0	1	0	0	1		
		1	0	1	0	0	0	1		
		1	0	1	1	0	1	1		
		1	1	0	0	0	0	1		
		1	1	0	1	0	1	1		
		1	1	1	0	0	1	1		
0	0	1	1	1	1	0	0	0	0	1
0	1	1	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	0	1
1	1	1	1	1	1	0	0	0	0	1

The α outputs may be expressed as follows:

$$\alpha^*_{44} = \left[a_{44}a^*_{44}b_{44}(ca)\right]\cdot\left[a_{43} \oplus b_{43}\right]$$

$$\alpha_{44} = \left[a_{44} \oplus b_{44} \oplus a^*_{44} \oplus ca\right] \vee \left[a_{44}a^*_{44}b_{44}(ca)\right] \cdot \left[a_{43} \oplus b_{43}\right]$$

$$\alpha_{43} = a_{43} \oplus b_{43} \oplus c_{43}$$

$$\alpha^*_{42} = \left[a_{43} \oplus b_{43}\right]c_{43} \vee a_{44}a^*_{44}b_{44}(ca)$$

$$c_{43} = \left[a_{44} \oplus a^*_{44}\right]\left[b_{44} \vee ca\right] \vee a_{44}a^*_{44}\left[\bar{b}_{44} \vee \overline{ca}\right] \vee \bar{a}_{44}b_{44}(ca)$$
$$\vee \left[a_{44}a^*_{44}b_{44}(ca)\right] \cdot \left[a_{43} \oplus b_{43}\right]$$

Note that unless $a_{44}a^*_{44}b_{44}(ca) = 1$, the special adder performs the same function as a standard base 4 pseudo adder.

The logic used in the physical realization of the low A special adder is given below. It is in agreement with the logic shown on LAD (D-1212).

$$\alpha^*_{44} = \left[a_{44}a^*_{44}b_{44}(ca)\right]\left[a_{43}\bar{b}_{43} \vee \bar{a}_{43}b_{43}\right]$$

$$\alpha_{44} = \left[(a_{44}\bar{a}^*_{44} \vee \bar{a}_{44}a^*_{44}) \oplus (b_{44} \vee ca)\ (\bar{b}_{44} \vee \bar{ca})\right] \vee \alpha^*_{44}$$

$$\alpha_{43} = \left[a_{43}b_{43} \vee \bar{a}_{43}\bar{b}_{43}\right] \equiv c_{43}$$

$$\alpha^*_{42} = \left[a_{43}\bar{b}_{43} \vee \bar{a}_{43}b_{43}\right] (c_{43})(\overline{KI}) \vee a_{44}a^*_{44}b_{44}(ca)$$

$$c_{43} = (a_{44}\bar{a}^*_{44} \vee \bar{a}_{44}a^*_{44})(b_{44} \vee ca) \vee a_{44}a^*_{44}(\bar{b}_{44} \vee \overline{ca}) \vee \bar{a}_{44}b_{44}(ca)$$
$$\vee a_{44}a^*_{44}b_{44}(ca)\left[a_{43}\bar{b}_{43} \vee \bar{a}_{43}b_{43}\right]$$

$$c_{42} = a^*_{42} \vee a_{42}b_{42}$$

$$\bar{d}_{42} = \bar{a}_{43} \vee \bar{b}_{43}$$

Note that $\overline{KI}$ does not appear in the expression for α^*_{44} since $b_{44} = 0$ when the MsA selector is set to KgA.

The expressions for the low S special adder are obtained by making the following substitutions: $\sigma = \alpha$, $x = c$, $y = d$, $s = a$, $t = b$, and $cs = ca$. The $\overline{KI}$ variable does not appear in the expression for σ^*_{42}.

3.1.6 One Digit Assimilators

The Q and R one digit assimilators are shown on the left side of the HQR drawing (D-1525). They each consist of a two input OR and a two input AND. Their function is to assimilate the stored carries which may appear in Q^*_0 and R^*_0. This is necessary for two reasons. If a left shift occurred without assimilation, the adder relation might be violated. If a right shift occurred without assimilation, Q^*_i and R^*_i memory elements would be needed.

A subtraction of M from S followed by a right shift maps σ^*_{42} into Q^*_0. (See section 3.1.2.10.) If the subtraction is case 3 or 4, $\sigma^*_{44} = 1$ and $Q^*_0 = 1$ following the shift. This is the only way to set $Q^*_0 = 1$. Similar statements are valid for R^*_0.

Since there is only one bit configuration for Q_{-1}, Q_0, and Q^*_0 which permits $Q^*_0 = 1$, assimilation is very simple. The output of the low S special adder is such that if $q^*_0 = 1$ following a left shift, then $q_0 = 1$ and $q_{-1} = 0$. Assimilation yields

$$\nu_{-1} = q_{-1} \vee q^*_0$$

and

$$\nu_0 = q_0 \overline{q^*_0}$$

For R we have

$$\rho_{-1} = r_{-1} r^*_0$$

and

$$\rho = r_0 \overline{r_0^*}$$

If the stored carry is zero in either case, the outputs of the assimilators agree with the respective bit positions of the Q and R registers.

3.1.7 Zero Detectors and Round-Off Logic

This logic appears on ZDR (D-1505). The S, R, and Q zero detectors are large OR trees. Their outputs $\overline{SZ}$, $\overline{RZ}$, and $\overline{QZ}$ are zero when the respective registers contain all zeros.

$$\overline{SZ} = s_{-o} \vee s_0 \vee s_1 \vee \text{----------} \vee s_{43} \vee s_{44}$$

$$\overline{RZ} = r_{-1} \vee r_0 \vee r_1 \vee \text{----------} \vee r_{43} \vee r_{44}$$

$$\overline{QZ} = \nu_{-1} \vee \nu_0 \vee q_{1\text{-}42}$$

where

$$q_{1\text{-}42} = q_1 \vee q_2 \vee \text{----------} \vee q_{41} \vee q_{42}$$

In order for $\overline{SZ}$ to be valid, the stored carries must be assimilated. The same would apply to $\overline{RZ}$ and $\overline{QZ}$ with respect to the carries or borrows stored in the 42^* and 44^* positions; however, these bits are always cleared to zero at the time $\overline{RZ}$ and $\overline{QZ}$ are inspected. A unit in R_0^* would not cause $\overline{RZ}$ to give a false indication, since there would also be a unit in R_0 as discussed in the preceding section. The formation of $\overline{QZ}$ is slightly different for two reasons. First of all, Q_{43} and Q_{44} are always cleared to zero during the decode step (GI) of any DC instruction and during K3 of the correct overflow sequence, (K), for SSC, ASC orders. (See sections 3.1.2.12 and 5.1.) In all cases, the $\overline{QZ}$ and round-off logic outputs are inspected by DC before any units have been gated

into Q_{43} and Q_{44}. This justifies the omission of q_{43} and q_{44} in the expression for q_{1-42}. The expression for $\overline{QZ}$ would be just as valid if q_{-1} and q_0 were used in place of ν_{-1} and ν_0 for the reasons discussed in the preceding section. The ν_{-1} and ν_0 signals are necessary inputs to the associated round-off logic, so it was convenient to use them for the formation of $\overline{QZ}$ as well.

Note that cable drivers and difference amplifiers are used to distribute the true and complement outputs of the zero detect logic. The control points of the DC sequences which use these outputs are indicated in the "Reference Logic" line of the output table.

The round-off logic is shown in the lower right corner of ZDR (D-1505). Its inputs are q_{1-42}, ν_{-1}, ν_0, $a_{44} \oplus a^*_{44}$, and ro. Its output is the unbiased round-off bit ρ.

$$\rho = (ro)(\nu_{-1}) \left[(a_{44} \oplus a^*_{44}) \vee \nu_0 \vee q_{1-42} \right]$$

As discussed earlier, $\rho = 0$ if the round-off status memory element RO is set to $\overline{RO}$, in which case its true output ro = 0. Prior to a round-off operation, DC must set RO causing ro = 1. In this condition, ρ may be 0 or 1 as defined above. It is added to the $44^{\underline{th}}$ position of the A register as a carry into the low end of the carry generator. (See section 3.1.8.)

The round-off carry and the stored carries of A are assimilated via the carry generator and A-adder when KgA is true. The rounded and assimilated result appears at the output of the A adder and is transferred to R in all cases. Following this transfer, DC sets $\overline{RO}$ to insure that $\rho = 0$.

The rules for unbiased round-off are stated as follows, where the binary point in Q is assumed to lie to the left of Q_{-1}.

$$\rho = 0 \quad \begin{cases} \text{if } Q < 1/2 \\ \text{or if } a_{44} \oplus a^*_{44} = 0 \text{ and } Q = 1/2 \end{cases}$$

$$\rho = 1 \begin{cases} \text{if } Q > 1/2 \\ \text{or if } a_{44} \oplus a_{44}^{*} = 1 \text{ and } Q = 1/2 \end{cases}$$

Since $q_0^{*} = q_0 = 1$ may occur, it is necessary to use the outputs of the one digit assimilator ν_{-1} and ν_0 together with q_{1-42} in determining the range of Q. The Boolean expression for ρ given above is seen to agree with these round-off rules. Note that $a_{44} \oplus a_{44}^{*}$ is formed in the low A special adder as $a_{44}a_{44}^{*} \vee \bar{a}_{44}a_{44}^{*}$.

3.1.8 Carry Generator

The carry generator logic is shown on CG (D-1099). It is used in conjunction with the A adder to assimilate the stored carries in A and the round-off bit. The inputs to the carry generator are ρ and all the bits in A except a_{-1}. Because of the manner in which the KgA gate signal is introduced, the fifth stage of the carry generator logic also includes a part of the MsA selector OR logic for the even-numbered bits. As a consequence, the b_i' (i even) are also inputs to the CG. This is illustrated in the upper left corner of the CG drawing. The expressions for the outputs of even and odd-numbered MsA selector-OR's are given below.

$$b_i = (MsA)m_i \vee (\bar{M}sA)\bar{m}_i \vee (2MsA)m_{i+1} \quad \Big\} \text{ i odd}$$

where

$$\left.\begin{aligned} b_i &= b_i' \vee (KgA)k_i \\ b_i' &= (MsA)m_i \vee (\bar{M}sA)\bar{m}_i \vee (2MsA)m_{i+1} \end{aligned}\right\} \text{ i even}$$

The flattened OR symbol is used when i is even to indicate that a non-standard OR circuit is employed. The k_i signals for i even are the true outputs

of the carry generator even though they are not shown explicitly on the CG drawing. Since $b_i = b_i' \vee (KgA)k_i$ is accomplished in the fifth stage of the CG, the outputs shown are the b_i. Note that $b_i = b_i'$ when KgA = 0, i.e., when the MsA selector is set to something other than KgA. However, when KgA = 1, $b_i' = 0$ and $b_i = k_i$ if i is even and $b_i = 0$ if i is odd. The k_i as used here does not equal K_i as shown on D-1099.

The K_0 output is used in the sign A and sign comparison logic which is discussed in section 3.1.11. The k_{-1} output is used in the high A bit path logic as discussed in section 3.1.2.8.

D. J. Wheeler designed the original carry generator as described in Report 92. Its use permits the assimilation of all the stored carries in A and the round-off bit, ρ, in one pass through the A adder with KgA = 1. To accomplish this, it is necessary to generate the carry, k_i, that is introduced in the least-significant position of each base 4 pseudo adder when all of the stored carry bits of A and/or the round-off bit are added to the non-carry bits of A. In order to have a carry into the $i^{\underline{th}}$ position (i even) there must be a stored carry bit or round-off bit and a non-carry bit in the $(i + j)^{\underline{th}}$ position (j even $\neq 0$) which is propagated into the $i^{\underline{th}}$ position by a continuous series of nonzero bits between i and i + j. When j = 0, the carry into the $i^{\underline{th}}$ position is zero; however, there may be a stored carry in that position. The general Boolean expression for k_i is given below. Note that a_i^* is not included.

$$k_i = a_{i+1}a_{i+2}a_{i+2}^* \vee a_{i+1}a_{i+2}a_{i+3}a_{i+4}a_{i+4}^* \vee \cdots \vee \left(\prod_{k=1}^{42-i} a_{i+k}\right)a_{42}^*$$
$$\vee \left(\prod_{k=1}^{44-i} a_{i+k}\right)(a_{44}^* \vee \rho)$$

Since the adder relation, $a_k^* a_{k+1} \equiv 0$, must always be satisfied, the terms in the general expression for k_i are orthogonal. That is, if $k_i = 1$, one

and only one of the AND terms on the right side of the equation has unit value. The $(a^{*}_{44} \vee \rho)$ term deserves special attention. Since the adder relation may be broken between A and Q, $a^{*}_{44} = \rho = 1$ may occur. A carry into the $43^{\underline{rd}}$ position will arise if either or both of a^{*}_{44} and ρ have unit value. Another unit is added to the $44^{\underline{th}}$ position if and only if $\rho = 1$. Hence, $k_{44} = \rho$.

The physical realization of the k_i functions is in terms of generate and propagate conditions as expressed below. For convenience, these functions are chassis-oriented and therefore span two base 4 positions in the first and fifth levels.

First-Level Logic

Carry Generation: $G_{45} = a^{*}_{44} \vee \rho = k_{44}$ (As shown on D-1099)

Carry Propagation: $P_{(i-3)-i} = a_{i-3}a_{i-2}a_{i-1}a_i$

Carry Generation: $G_{(i-3)-i} = a^{*}_{i-4} \vee a_{i-3}a_{i-2}a^{*}_{i-2}$

where $i = 4q$ and $1 \leq q \leq 11$.

Second-Level Logic

$$G_{33-45} = G_{33-36} \vee P_{33-36}G_{37-40} \vee P_{33-36}P_{37-40}G_{41-44} \vee P_{33-36}P_{37-40}P_{41-44}G_{45}$$

$$P_{21-32} = P_{21-24}P_{25-28}P_{29-32}$$

$$G_{21-32} = G_{21-24} \vee P_{21-24}G_{25-28} \vee P_{21-24}P_{25-28}G_{29-32}$$

$$P_{9-20} = P_{9-12}P_{13-16}P_{17-20}$$

$$G_{9-20} = G_{9-12} \vee P_{9-12}G_{13-16} \vee P_{9-12}P_{13-16}G_{17-20}$$

Third-Level Logic

$$G_{21-45} = G_{21-32} \vee P_{21-32}G_{33-45}$$

$$G_{9-45} = G_{9-20} \vee P_{9-20}G_{21-32} \vee P_{9-20}P_{21-32}G_{33-45}$$

Fourth-Level Logic

$$G_{41-45} = G_{41-44} \vee P_{41-44}G_{45}$$

$$G_{37-45} = G_{37-40} \vee P_{37-40}G_{41-44} \vee P_{37-40}P_{41-44}G_{45}$$

$$G_{29-45} = G_{29-32} \vee P_{29-32}G_{33-45}$$

$$G_{25-45} = G_{25-28} \vee P_{25-28}G_{29-32} \vee P_{25-28}P_{29-32}G_{33-45}$$

$$G_{17-45} = G_{17-20} \vee P_{17-20}G_{21-45}$$

$$G_{13-45} = G_{13-16} \vee P_{13-16}G_{17-20} \vee P_{13-16}P_{17-20}G_{21-45}$$

$$G_{5-45} = G_{5-8} \vee P_{5-8}G_{9-45}$$

$$G_{1-45} = G_{1-4} \vee P_{1-4}G_{5-8} \vee P_{1-4}P_{5-8}G_{9-45}$$

Fifth-Level Logic

The following set of carry generate signals have been formed in the 1st, 2nd, 3rd, and 4th levels. $K_i \equiv G_{(i+1)-45}$ where $i = 4q$ and $0 \leq q \leq 11$. To obtain the carry into the i^{th} position, k_i, where $i = 4q$ and $0 \leq q \leq 10$, we form

$$k_i = \overline{a}_i^* K_i \quad \text{and} \quad k_{44} = \rho$$

Likewise, to obtain the carry into the $(i - 2)^{nd}$ position, k_{i-2}, where $i - 4q$ and $1 \leq q \leq 11$, we form

$$k_{i-2} = a_{i-1}a_iK_i$$

The significance of these expressions becomes apparent if we examine two special cases of the general carry equation. We know that k_{40} and k_{42} may be expressed as follows:

$$k_{40} = a_{41}a_{42}\overset{*}{a}_{42} \vee a_{41}a_{42}a_{43}a_{44}(\overset{*}{a}_{44} \vee \rho)$$

$$k_{42} = a_{43}a_{44}(\overset{*}{a}_{44} \vee \rho)$$

The implicit outputs, k_{40} and k_{42}, can be written in terms of K_{40} and K_{42} respectively.

$$k_{40} = \overline{\overset{*}{a}}_{40}K_{40} = \overline{\overset{*}{a}}_{40}(G_{41-44} \vee P_{41-44}G_{45})$$

$$k_{40} = \overline{\overset{*}{a}}_{40}\left[\overset{*}{a}_{40} \vee a_{41}a_{42}\overset{*}{a}_{42} \vee a_{41}a_{42}a_{43}a_{44}(\overset{*}{a}_{44} \vee \rho)\right]$$

Since $\overset{*}{a}_{40}a_{41} \equiv 0$, we have $\overline{\overset{*}{a}}_{40}a_{41} \equiv a_{41}$. It follows that

$$k_{40} = a_{41}a_{42}\overset{*}{a}_{42} \vee a_{41}a_{42}a_{43}a_{44}(\overset{*}{a}_{44} \vee \rho)$$

In terms of K_{42} we have

$$k_{42} = a_{43}a_{44}K_{44} = a_{43}a_{44}G_{45} = a_{43}a_{44}(\overset{*}{a}_{44} \vee \rho)$$

It is easy to generalize these arguments to show that $k_i = \overline{\overset{*}{a}}_iK_i$ or $k_{i-2} = a_{i-1}a_iK_i$ for all even i in the range $0 \leq i \leq 42$.

The KgA gate and b_i' signals are also introduced in the fifth level. The principal outputs of the carry generator are expressed as follows:

$$b_i = b_i' \vee k_i(KgA)$$

$$b_{i-2} = b'_{i-2} \vee k_{i-2}(KgA)$$

where k_i and k_{i-2} are defined above. Because of the action of the MsA selector, $b'_i = 0$ when KgA = 1, so $b_i = k_i$ when KgA = 1.

The outputs K_0 and k_{-1} are used in the sign A logic and high A bit path logic respectively. The $K_0 \equiv G_{1-45}$ signal is formed as indicated above, while the k_{-1} signal is generated as

$$k_{-1} = (a_0 K_0)KgA$$

The choice of k_{-1} as a symbol to represent this signal is unfortunate since in general $K_i \equiv G_{(i+1)-45}$ for i even. Note that when KgA = 1, k_{-1} represents the carry into the (-1) position.

3.1.9 The α Logic

As shown on the right side of the DPα drawing (D-1507), the α logic is fed by the α_{-1}, α_0, α_1, α_2, α_3 and α_4 outputs of the A adder. Its outputs are αov, αnz, nα, and nα' with their complements. These outputs are distributed to the various control sequence steps as indicated. It is important to note that these outputs are meaningful if and only if KgA = 1 and sufficient time has elapsed to permit the carry generator and A adder to react. Under these conditions, the A adder output, α, represents the assimilated value of A with or without round-off. The Boolean expressions for the four outputs are given below.

$$\alpha ov = \alpha_{-1}\bar{\alpha}_0 \vee \bar{\alpha}_{-1}\alpha_0$$

If αov = 1, α is overflowed Mod 2, i.e., $\alpha_{-1} \neq \alpha_0$.

$$\alpha nz = \overline{[(\bar{\alpha}_0\bar{\alpha}_1\bar{\alpha}_2)(\bar{\alpha}_3\bar{\alpha}_4)]}$$

If $\alpha nz = 1$, α is nonzero since at least one of the outputs α_0, α_1, α_2, α_3, and α_4 has unit value.

$$n\alpha = \overline{[\alpha_0\alpha_1\alpha_2 \vee \bar{\alpha}_0\bar{\alpha}_1\bar{\alpha}_2]}$$

If $n\alpha = 1$, $-\frac{7}{4} \leq \alpha < -\frac{1}{4}$ or $\frac{1}{4} \leq \alpha < \frac{7}{4}$. If $\alpha ov = 0$ when $n\alpha = 1$, then $-1 \leq \alpha < -\frac{1}{4}$ or $\frac{1}{4} \leq \alpha < 1$ which corresponds to the normalized range of α except that $\alpha = 0$ is not included. Therefore, even though $n\alpha$ is associated with the normalized α condition, it does not always imply this condition.

$$n\alpha' = \alpha_0\alpha_1\alpha_2(\bar{\alpha}_3 \vee \bar{\alpha}_4) \vee \bar{\alpha}_0\bar{\alpha}_1\bar{\alpha}_2(\alpha_3 \vee \alpha_4)$$

If $n\alpha' = 1$, then α lies within one of the following ranges:

$$-\frac{31}{16} \leq \alpha < -\frac{28}{16} \qquad -\frac{1}{4} \leq \alpha < -\frac{1}{16}$$

$$\frac{1}{16} \leq \alpha < \frac{1}{4} \qquad \frac{28}{16} \leq \alpha < \frac{31}{16}$$

If $\alpha ov = 0$ when $n\alpha' = 1$, then $-\frac{1}{4} \leq \alpha < -\frac{1}{16}$ or $\frac{1}{16} \leq \alpha < \frac{1}{4}$. Therefore, the condition $(\overline{\alpha ov})(n\alpha') = 1$ implies that α will be normalized after one base 4 shift to the left.

In many cases when $n\alpha$ and $n\alpha'$ are sampled, it is known apriori that α is not overflowed. This is true in both the normalized (R) and shift (F) sequences as a consequence of leaving the accumulator in a non-overflowed state at the end of every Delayed Control instruction. Note that αov is not used in either the (R) or (F) sequences. In the store (S) sequence, αov appears in conjunction with $n\alpha$ because round-off may cause overflow. These considerations are also discussed in the sections which deal with the (R), (F), and (S) control sequences.

3.1.10 The A, S, M and R Normalization Logic

These individual blocks of logic are shown on the upper half of the SEC drawing (D-1527). The Boolean expression for each is given below.

$$na = \overline{a_0 a_1 a_2 \vee \bar{a}_0 \bar{a}_1 \bar{a}_2}$$

$$ns = \overline{s_0 s_1 s_2 \vee \bar{s}_0 \bar{s}_1 \bar{s}_2}$$

$$mn = \overline{m_0 m_1 m_2 \vee \bar{m}_0 \bar{m}_1 \bar{m}_2}$$

$$nr = \overline{r_0 r_1 r_2 \vee \bar{r}_0 \bar{r}_1 \bar{r}_2}$$

If any of these expressions has unit value, the contents of the corresponding register must be in the range $-\frac{7}{4} \leq f < -\frac{1}{4}$ or $\frac{1}{4} \leq f < \frac{7}{4}$. If it is also known that the register does not hold a number which is overflowed Mod 2, then f must be in the normalized range: $-1 \leq f < -\frac{1}{4}$ or $\frac{1}{4} \leq f < 1$.

The na and ns outputs are valid only when the stored carries in the A and S registers are assimilated. These signals and their complements are used in the shift (S) sequence exclusively. Because the accumulator is left in a non-overflowed state at the end of every Delayed Control instruction, the condition that na = 1 or ns = 1 implies that the number contained in the A or S register is in the normalized range.

The nm and nr signals and their complements are used only in the division (D) sequence. (See section 5.13.)

The DIV and NDV orders place the divisor in M as the initial operand. Therefore, it cannot be overflowed in the fractional part. It follows that if mn = 1, the divisor is normalized. The VID order obtains a divisor by normalizing the accumulator and then rounding. As a result, the fractional part of the divisor which is placed in M is either zero or in the range $\frac{1}{4} \leq |f| \leq 1$.

Thus M may contain a $f = -\frac{1}{4}$ or +1. If $f = -\frac{1}{4}$, nr = 0; while if f = +1, nm = 1 since f looks like -1 to the mn logic. This situation does not cause trouble in the VID because the $\bar{x}$ signal masks nm at the D2 control point. (See DC Flow Chart D-1128.)

For any divide order the nr signal is used to determine when a "normalized" quotient has been formed. The condition nr = 1 causes an exit from the iterative loop (control points D10 and D11). At the time of exit, R may or may not contain an assimilated representation of the unrounded quotient. There may be an effective stored <u>carry</u> in the "R^*_{40}" position which would be absorbed as soon as a negative quotient digit (bits β_{42}, ρ_{43}, and ρ_{44}) is generated and placed in R^*_{42}, R_{43}, and R_{44}. The "R^*_{40}" memory element does not exist; but if it did, it would contain a carry bit under certain conditions during the generation of a quotient. If "R^*_{40}" contains a carry and a nonzero negative quotient digit does not arise before the division terminates, the carry remains and must be assimilated to obtain the true representation of the unrounded quotient. (A detailed explanation of quotient generation is given in sections 3.1.17 and 5.13.) The point of interest here is the fact that the bit pattern in R_0, R_1 and R_2 may change if the effective carry in "R^*_{40}" were assimilated. This change would also cause nr to change in the cases shown below.

R Bit Position :	-1	0.	1	2	3	4	...	...	39	40	"40*"	41	42	42*	43	44
$R = -\frac{1}{4}$:	1	1.	1	0	1	1	...	1...	1	1	1	0	0	1	0	0
$R = +\frac{1}{4}$:	0	0.	0	0	1	1	...	1...	1	1	1	0	0	1	0	0
R = +1 :	0	0.	1	1	1	1	...	1...	1	1	1	0	0	1	0	0

R^*_{42} contains a stored <u>borrow</u> in these cases, but it must be ignored since both R_{43} and R_{44} contain zeros, which means the last quotient digit generated was a negative zero.

In the first and third cases, nr = 1, but the unrounded quotient is not normalized since R = $-\frac{1}{4}$ or +1. In the second case nr = 0, but the unrounded quotient is in fact normalized, since R = $+\frac{1}{4}$. Since nr = 0 another pass through the divide loop occurs. If a negative quotient digit is not inserted during this pass, case 3 will arise. None of these cases causes any difficulty, since $-\frac{1}{4}$ and +1 are both valid unrounded quotients. The effect of rounding is considered in section 5.13. In all other cases, nr gives a true indication of whether or not the unrounded quotient is normalized.

3.1.11 Sign A and Sign Comparison Logic

This logic is shown in the upper right corner of SEC (D-1527). As its name implies, it is used to obtain the true sign of A and to compare that sign with the sign of M. Since the K_0 output of the carry generator is used, the MsA selector does not have to be set to KgA to obtain the true sign of A. This is of no consequence as far as the use of the Sign A signal is concerned; however, it is a necessary requirement for generating the sign comparison signal, γ, during the divide loop.

$$\text{Sign A} = \overline{\left[\overline{a}_{-1} \oplus a_0 K_0\right]} = a_{-1} \oplus a_0 K_0$$

The true sign of A is always taken as the assimilated value of a_{-1}. Since $a_0 K_0$ represents the carry into the A_{-1} position under assimilation (see section 3.1.8), Sign A can be expressed as shown above.

$$\gamma = (\text{Sign A}) m_{-1} \vee (\overline{\text{Sign A}})\, \overline{m}_{-1}$$

The sign of M is always determined by the contents of M_{-1}. If A and M have like signs, γ = 1. Since the sign comparison signal, γ, is used only during division, the signs of the partial remainder in A and the divisor in M

agree if $\gamma = 1$ and disagree otherwise. An explanation of the use of γ is given in sections 3.1.17 and 5.13.

3.1.12 The Q and R Half-Subtractors

The Q and R half-subtractor logic is shown in the lower left corner of SEC (D-1527). These half-subtractors are used only in case 3 of floating addition. In this case, the exponent of the augend (subtrahend) is larger than the exponent of the addend (minuend) so the fractional portion of the augend (subtrahend), AQ, must be circularly left shifted until the exponents agree. The accumulator is assimilated during the first transfer from A to S in this shift process. This places the most significant bits of the augend (subtrahend) in the least significant portion of the Q or R register at the time addition (subtraction) occurs.

As a consequence of this addition (subtraction), a borrow or carry or neither may be generated in the (-1) position. If a borrow is generated, the Q and R half-subtractors are used to propagate it into the most significant bits of the sum (difference) during the circular right shift into AQ. The Q^*_{44} and R^*_{44} memory elements alternately hold the borrow during propagation. If a carry arises, the high A and S special adders (section 3.1.4) propagate it into the most significant bits of the sum (difference) during the circular right shift into AQ. A^*_0 and S^*_0 alternately hold this carry during propagation.

Whenever a borrow is not generated at the time of addition (subtraction), the α'_{-3}, α'_{-2} and σ'_{-3}, σ'_{-2} outputs of the Q and R half-subtractors are simply copies of q_{43}, q_{44} and 4_{43}, r_{44}, and the Q^*_{44} and R^*_{44} memory elements are always set to zero. In any case the α'_{-3} and α'_{-2} (σ'_{-3} and σ'_{-2}) outputs of the Q (R) half-subtractor are used to set S_{-1} and S_0 (A_{-1} and A_0) during the circular right shift. (See sections 3.1.2.8 and 3.1.2.6 for bit path details.)

Q Half-Subtractor Logic

$$m\alpha_{-2} = \overset{*}{q}_{44}(\text{OMsA}) \vee \bar{a}_{-1}b_{-1}(\overline{\text{OMsA}})$$

$$\alpha'_{-3} = \overline{q_{43} \textcircled{\equiv} (\bar{q}_{44})(m\alpha_{-2})} = q_{43} \oplus \bar{q}_{44}(m\alpha_{-2})$$

$$\alpha'_{-2} = \overline{q_{44} \textcircled{\equiv} m\alpha_{-2}} = q_{44} \oplus m\alpha_{-2}$$

$$\overset{*}{R}_{44} = (\text{cr})\,(\overline{q_{43}})(\overline{q_{44}})(m\alpha_{-2})$$

R Half-Subtractor Logic

$$m\sigma_{-2} = \overset{*}{r}_{44}(\text{OMsS}) \vee \bar{s}_{-1}t_{-1}(\overline{\text{OMsS}})$$

$$\sigma'_{-3} = \overline{r_{43} \textcircled{\equiv} (\overline{r_{44}})(m\sigma_{-2})} = r_{43} \oplus (\bar{r}_{44})(m\sigma_{-2})$$

$$\sigma'_{-2} = \overline{r_{44} \textcircled{\equiv} m\sigma_{-2}} = r_{44} \oplus m\sigma_{-2}$$

$$\overset{*}{Q}_{44} = (\text{cr})(\bar{r}_{43})(\bar{r}_{44})(m\sigma_{-2})$$

These expressions are best understood by analyzing a case 3 addition (subtraction) in the (-1) column. Assume that at the end of the circular left shift, the most significant bits of the augend (subtrahend) lie in the least significant bits of the Q(R). The addend (minuend) in M is added (subtracted) to that portion of the augend (subtrahend) that is in A(S). The result is circularly right shifted one base 4 position into SR (AQ). This is generally only the first of many circular right shifts. At the time of addition (subtraction),

the augend (subtrahend) is always assimilated in A(S) because KgA = 1 during the first circular left shift. (This is clearly necessary since Q and R cannot hold stored carries.) The $a_{-1}(s_{-1})$ bit has positive weight in this case. The $b_{-1}(t_{-1})$ bit has negative weight as usual. Let $c_{-1}(x_{-1})$ be the carry into the $a_{-1}(s_{-1})$ position. Let $\alpha_{-1}(\sigma_{-1})$ and $\alpha^*_{-2}(\sigma^*_{-2})$ be the sum and stored carry bits as usual. Let $\beta_{-2}(\tau_{-2})$ represent the borrow from the -2 position. Expressions for $\alpha_{-1}(\sigma_{-1})$, $\alpha^*_{-2}(\sigma^*_{-2})$, and $\beta_{-2}(\tau_{-2})$ in terms of $a_{-1}(s_{-1})$, $b_{-1}(t_{-1})$, and $c_{-1}(x_{-1})$ can be derived from the truth table shown below. Keep in mind that $\alpha^*_{-2}(\sigma^*_{-2})$ are formed as in any standard base 4 pseudo adder. (See sections 3.1.3 and 3.1.4.)

Weights:	+2	-2	+2	+2	+4	-4
Variables:	$a_{-1}(s_{-1})$	$b_{-1}(t_{-1})$	$c_{-1}(x_{-1})$	$\alpha_{-1}(\sigma_{-1})$	$\alpha^*_{-2}(\sigma^*_{-2})$	$\beta_{-2}(\tau_{-2})$
	0	0	0	0	0	0
	0	0	1	1	0	0
	0	1	0	1	0	1
	0	1	1	0	1	1
	1	0	0	1	0	0
	1	0	1	0	1	0
	1	1	0	0	0	0
	1	1	1	1	0	0

$$\alpha_{-1} = a_{-1} \oplus b_{-1} \oplus c_{-1} \qquad \sigma_{-1} = s_{-1} \oplus t_{-1} \oplus x_{-1}$$

$$\alpha^*_{-1} = (a_{-1} \oplus b_{-1})c_{-1} \qquad \sigma^*_{-2} = (s_{-1} \oplus t_{-1})x_{-1}$$

$$\beta_{-2} = \bar{a}_{-1}b_{-1} \qquad \tau_{-2} = \bar{s}_{-1}t_{-1}$$

Note that the equations for $\alpha_{-1}(\sigma_{-1})$ and $\alpha^*_{-2}(\sigma^*_{-2})$ agree with those given in section 3.1.4.

During the first circular right shift into SR (AQ) the MsA (MsS) selecto

is set to $\overline{\text{MsA}}$ ($\overline{\text{MsS}}$) or MsA (MsS) while the MsS (MsA) selector is set to OMsS (OMsA). On the second circular right shift, MsA (MsS) is set to OMsA (OMsS). Thus, if $q_{43} = q_{44} = 0$ ($r_{43} = r_{44} = 0$) during the first circular right shift into SR (AQ), the borrow is propagated and must be stored in R^*_{44} (Q^*_{44}). If $r_{43} = r_{44} = 0$ and $R^*_{44} = 1$ ($q_{43} = q_{44} = 0$ and $Q^*_{44} = 1$) during the second circular right shift into AQ (SR), then the borrow is again propagated and must be stored in Q^*_{44} (R^*_{44}).

It follows that a unit $m\alpha_{-2}$ ($m\sigma_{-2}$), must be subtracted from q_{43}, q_{44} (r_{43}, r_{44}) if $\beta_{-2} = 1$ ($\tau_{-2} = 1$) and $\overline{\text{OMsA}} = 1$ ($\overline{\text{OMsS}} = 1$) or $q^*_{44} = 1$ ($r^*_{44} = 1$) and OMsA = 1 (OMsS = 1). The corresponding Boolean expressions for $m\alpha_{-2}$ and $m\sigma_{-2}$ were given above where $\beta_{-2} = \bar{a}_{-1}b_{-1}$ and $\tau_{-2} = \bar{s}_{-1}t_{-1}$. Note that $\bar{a}_{-1}b_{-1}$ ($\overline{\text{OMsA}}$) $\left(\bar{s}_{-1}t_{-1}(\overline{\text{OMsS}})\right)$ determines $m\alpha_{-2}$ ($m\sigma_{-2}$) on the first circular right shift into SR (AQ); and that q^*_{44}(OMsA) (r^*_{44}(OMsS)) determines $m\alpha_{-2}$($m\sigma_{-2}$) on every odd circular right shift after the first. The equations for the half subtractor logic can be derived from the following truth table.

$q_{43}(r_{43})$	$q_{44}(r_{44})$	$m\alpha_{-2}(m\sigma_{-2})$	$\alpha_{-3}(\sigma_{-3})$	$\alpha_{-2}(\sigma_{-2})$	$R^*_{44}(Q^*_{44})$
0	0	0	0	0	0
0	0	1	1	1	1
0	1	0	0	1	0
0	1	1	0	0	0
1	0	0	1	0	0
1	0	1	0	1	0
1	1	0	1	1	0
1	1	1	1	0	0

The cr signal, i.e., the true output of the CR status memory element, is included in the expression for setting R^*_{44} and Q^*_{44} in order to clear these memory elements to zero when they are not in use. The CR memory element is turned on (i.e., set to "1") at the beginning of a circular right shift and

turned off when the shift is completed. The circular left and right shift facility is used only during case 3 of floating addition. Therefore, R^*_{44} and Q^*_{44} are always set to zero except when a borrow is being propagated during a circular right shift following a case 3 addition. (See section 5.1.)

3.1.13 The Carry-Borrow Logic

This logic appears in the lower right corner of SEC (D-1527). In case 4 of addition, it is used to detect and temporarily hold a carry or borrow into the most significant half of the sum or difference. In division, it is used to detect and hold the result of a sign comparison between the divisor and the final partial remainder. (The final partial remainder is not the same as the remainder which is left in R at the end of any division. For details see section 5.13.)

The input to the CB memory element will be called CB even though it is not shown as such on the logic. As usual, the output of CB is called cb. In the following equation δ_2 is the true output of the Δ_2 status memory element. Δ_2 is set to "1" during the first pass through control point D10 in divide. Its true output, δ_2, affects the carry-borrow logic as shown below during D12, and the resulting output is gated into CB during D13. (See Delayed Control Flow Chart.) During D14, Δ_2 is set to "0" such that $\delta_2 = 0$. $\bar{\Delta}_2$ is true ($\delta_2 = 0$) during all other control sequences including addition.

$$CB = \alpha_{-1}\bar{m}_{-1}(\bar{m}_0 \vee \delta_2) \vee \bar{\alpha}_{-1}m_{-1}(m_0 \vee \delta_2)$$

The significance of this logic will be discussed first in relation to case 4 floating addition. In this case, the CB output is gated into CB during A11. (See the DC Flow Chart D-1128.) If cb = 1 after this gating operation, it is interpreted by the conditional logic in A12 as either a carry or borrow

depending on the sign of M. If cb = 0, it is interpreted as neither a carry or a borrow, and A12 causes nothing to be added to the least significant bit of the most significant half of the sum.

The carry-borrow logic was suggested by R. H. Farrell. To understand its function in case 4 addition, it is necessary to look at the way this particular case of addition is executed. In case 4 addition, the exponent of the augend (subtrahend) is so much greater than the exponent of the addend (minuend) that the latter must be added (subtracted) to the least significant part of the former. To accomplish this, the most significant half of the augend (subtrahend) is placed in Q, and the least significant half is placed in R with zeros in S. (See the DC Flow Chart and section 5.5.) The addend (minuend) in M is then added (subtracted) to the zeros in S and the result gated straight down (without any base 4 shift) into A. As a consequence, the contents of A are then always added to the least significant half of the augend (subtrahend) which is gated from R to M during A7. Prior to the addition, it may be necessary to shift the contents of A right until its original exponent difference has been reduced to zero. The addition of M therefore may be to either A or S.

At the time of addition, the quantity in A or S is signed and may lie anywhere in the range: $-1 \leq f \leq +1$. Note that +1 must be included because the minuend may have been -1. The quantity in M is always positive or zero and may lie anywhere in the range: $0 \leq m < 4$. The point in M is taken between M_1 and M_0 as usual. Both M_{-1} and M_0 have positive weight since they were originally Q_{-1} and Q_0; i.e., the $45^{\underline{th}}$ and $46^{\underline{th}}$ bits of the augend (subtrahend). The sum A + M or S + M is always placed in A during the A10 control step prior to leaving the add loop (A9-A10). The limits on the quantities added insure that the sum in A lies in the range: $-1 \leq f < 5$. It is clear that a carry must be propagated into the most significant portion of the augend (subtrahend) if $4 \leq f < 5$. Likewise, it is clear that a borrow must be propagated if $-1 \leq f < 0$.

It is possible to establish whether the sum lies in either of these ranges by examining the sign of the sum and the range of M. To determine the sign of the sum which is in A, we make use of the fact that the MsA selector is set to KgA during A10 in the final pass through the add loop (A9-A10). This means that the assimilated value of A appears at the output of the A adder, α, as control passes to A11. The true sign of A, i.e., the sum, is given by α_{-1} at that time. If $\alpha_{-1} = 0$, the sum, α, must lie in one of the following ranges: $0 \leq \alpha < 2$ or $4 \leq \alpha < 5$. We determine which range by knowing the range of M. In order for the sum to lie in the higher range, it is necessary for $3 \leq m < 4$. Note that it is never possible for the sum to lie in the lower range when $3 \leq m < 4$. Thus, if $\bar{\alpha}_{-1} m_{-1} m_0 = 1$, a carry must be propagated into the most significant part of the augend (subtrahend) which then becomes the most significant part of the sum.

If $\alpha_{-1} = 1$, the sum, α, must lie in one of the following ranges: $-1 \leq 2 < 0$ or $2 \leq \alpha < 4$. As before, we determine which range applies by knowing the range of M. In order for the sum to be in the lower range, M must be in the range $0 \leq m < 1$. Furthermore, if m is in this range, the sum cannot lie in the higher range. Therefore is $\alpha_1 \bar{m}_1 \bar{m}_0 = 1$, a borrow must be propagated into the most significant part of the augend (subtrahend) which then becomes the most significant part of the sum.

We note that a carry or borrow must be propagated if $\alpha_{-1} \bar{m}_{-1} \bar{m}_0 \vee \bar{\alpha}_{-1} m_{-1} m_0$ has unit value. This expression is identical to the expression for CB when $\delta_2 = 0$, which is always true except during division.

Even though CB is stored in the CB memeory element in A11, it is still necessary to decide whether cb = 1 is a carry or a borrow. This is done by the conditional logic at the A12 control point. As shown on the Delayed Control Flow Chart, the MsS selector is set to CS if $(cb)m_0 = 1$; to $M\bar{M}sS$ if $(cb)\bar{m}_0 = 1$; and left at OMsS if cb = 1. From the above analysis, it is clear that cb = 1 only when $m_{-1} = m_0$. Furthermore, cb = 1 must be interpreted as a carry if

$m_0 = 1$ and as a borrow if $m_0 = 0$. If cb = 0, neither a carry nor a borrow is propagated. It follows that a carry must be added in the $44^{\underline{th}}$ position by setting CS if $(cb)m_0 = 1$; and a borrow must be added in that position by adding a field of units to S via $\overline{M}\overline{M}sS$, if $(cb)\overline{m}_0 = 1$.

In division, the carry-borrow logic compares the signs of the divisor and the final partial remainder as described earlier. The result of this comparison, CB, is stored in the CB memory element during D13. The cb signal is used in the conditional logic of D14 where the remainder is corrected for quotient round-off and in D17 where the quotient round-off is actually performed.

We note that $CB = \alpha_{-1} \oplus m_{-1}$ when $\delta_2 = 1$. As such, CB has opposite parity with respect to the sign comparison output, γ, that is used during the divide loop (D10-D11). (See section 3.1.11.) Because A changes, the γ signal could not be used in determining quotient round-off unless it were stored in some manner. Instead of storing γ, it was convenient to convert the carry-borrow logic to sign comparison logic using δ_2 and then store the output as in addition.

The final partial remainder is placed in A during D12. (See the DC Flow Chart, D-1128.) During this same control step, the MsA selector is set to KgA such that the A adder output, α, represents the assimilated value of A after sufficient time has been allowed for the carry generator and adder logic to react. The true sign of the final partial remainder in A is thus given by α_{-1}. (The partial remainders in A always lie in the range: $-2 \leq a < \frac{5}{3}$. See section 3.1.16.) The divisor, d, may lie in either of the following ranges: $-1 \leq d \leq -\frac{1}{4}$ or $\frac{1}{4} \leq d \leq +1$. A divisor of +1 may arise while executing VID. If the accumulator equals $1 - 2^{-44}$ after normalization, then round-off on the way to R during D2 may yield a +1 divisor. Therefore, the true sign of the divisor in M is given by m_{-1} in all cases.

If $\alpha_{-1} \neq m_{-1}$, a unit is stored in CB. The significance of the cb

signal in determining the quotient round-off and the prior correction of the remainder is discussed in section 4.13.

3.1.14 The θ Decoder

Although in a strict sense, the θ decoders are not a part of the MAU logic, they are considered here for convenience and because they are the simplest of the decoders which set the MsA and MsS selector mechanisms. They are used only in the add (A) and clear add (B) sequences. The logic is shown on the right side of μΘD (D-1506). The outputs of the Θ_1 and Θ_2 status memory elements are decoded to form set signals for the MsS and MsA selectors as described below in negative logic.

θMsA Decoder Logic

$$\text{MsA} - \theta = (\bar{\theta}_1 \vee \bar{\theta}_2) \vee \theta\text{MsA}$$

$$\bar{\text{M}}\text{sA} - \theta = (\theta_1 \vee \theta_2) \vee \theta\text{MsA}$$

θMsS Decoder Logic

$$2\text{MsS} - \theta = (\bar{\theta}_1 \vee \bar{\theta}_2) \vee \theta\text{MsS}$$

$$\text{MsS} - \theta = (\bar{\theta}_1 \vee \theta_2) \vee \theta\text{MsS}$$

$$\bar{\text{M}}\text{sS} - \theta = (\theta_1 \vee \theta_2) \vee \theta\text{MsS}$$

$$\overline{2\text{M}}\text{sS} - \theta = (\theta_1 \vee \bar{\theta}_2) \vee \theta\text{MsS}$$

The θ status memory elements are set during some control step, usually decode (G1), prior to the step that uses their outputs to set the MsA or MsS selector mechanisms.

θ Outputs		Operation
θ_1	θ_2	
0	0	Subtract M
1	0	Add M
0	1	Subtract 2M
1	1	Add 2M

As indicated earlier, the requests from a control point are in the form of zero (negative voltage) outputs. All selector mechanisms respond to zero inputs. The θ decoder logic, as well as the μ and ρ decoder logic described below, must generate zero outputs in the active stage. For example, to set the MsA selector to $\overline{\text{M}}$sA, via the θ decoder, it is necessary to make $\overline{\text{M}}$sA - θ = 0. This can only occur when $\theta_1 = \theta_2 = 0$ and a control point request for θMsA is present, i.e., when θMsA = 0. As long as θMsA = 1, the outputs of the θMsA decoder have no effect on the MsA selector mechanism. (See section 4.2.) Similar statements can be made with regard to the θMsS decoder.

3.1.15 The μ Decoder

Most of this logic is shown in the left and center sections of $\mu\theta$D (D-1506). The remainder is shown as part of the input logic to Q^*_{42} and R^*_{42} on LQR (D-1524). The principal outputs of the μ decoder set the MsA and MsS selectors during multiplication.

A brief description of the multiplication instruction is included at this point to clarify the discussion of the μ decoder logic. A detailed description is given in section 5.12.

At the beginning of the multiply instruction, the multiplier is in AQ and the multiplicand is in M. The normalization sequence (R) is used to shift the multiplier left until it is normalized or Q = 0, whichever occurs

first. It is then rounded and transferred to R while zeros are placed in S. The two least significant bits of the rounded multiplier, α_{43} and α_{44}, are recoded by the μMsS decoder logic on their way to R_{43} and R_{44}. The output of this logic sets the MsS selector to one of the following: 2MsS, MsS, 0MsS, or $\overline{M}$sS. If $\overline{M}$sS is set during this initial step, a unit is also gated into R^*_{42} as shown on LQR (D-1524).

On the following step, the next two multiplier bits, r_{41} and r_{42}, and the mode bit, r^*_{42}, are recoded by the μMsA decoder logic. The output of this logic is used to set the MsA selector to one of the following: 2MsA, MsA, 0MsA, or $\overline{M}$sA. Another output on LQR is used to set Q^*_{42} during the down right transfer of S + μM,R into A,Q. After the transfer, $\frac{1}{4}$(S + μM,R) appears in A,Q. If $r^*_{42} = 1$, the r_{41} and r_{42} bits are modified by the borrow-subtractor logic (see section 3.1.17) before they are gated into Q_{43} and Q_{44}. This does not affect the setting of the MsA selector. If $2r_{41} + r_{42} + r^*_{42} \geq 3$, Q^*_{42} is set to 1 by the μMsA decoder logic.

On the next step, q_{41}, q_{42} and q^*_{42} are recoded by the μMsS logic. The outputs of this logic set $R^*_{42} = 1$ if $2q_{41} + q_{42} + q^*_{42} \geq 3$, and set the MsS selector to one of the four options listed above. During this step, $\frac{1}{4}$(A + μM,Q) is placed in S,R. The q_{41} and q_{42} multiplier bits are transferred to R_{43} and R_{44} without modification. Even if modification occurred during transfer, it would not affect the multiplication.

This process continues until the two sign bits of the multiplier have been sensed by the μMsS decoder logic in Q_{41} and Q_{42}. The up right transfer places $\frac{1}{4}$(A + μM,Q) in S,R with R^*_{42} set to 0. The following straight down transfer places S + μM,R in A,Q as the double length product. $Q^*_{42} = 0$ while Q_{43} and Q_{44} contain the sign bits of the multiplier at this point. By setting $R^*_{42} = 0$ during the last transfer into R, the borrow-subtractor is prevented from

modifying the two least significant bits of the product, r_{41} and r_{42}, as they are gated straight down into Q_{41} and Q_{42}.

The multiplication algorithm is based on a recoded multiplier. The recoding is accomplished by the μ decoder logic. The primary reason for recoding the multiplier is to avoid the need for 3M as a multiple of the multiplicand. Such a multiple would be necessary if we simply examined two bits of the multiplier at each step. In order to emphasize this point, the M selector settings required in each case without recoding are listed below.

Multiplier Bits	M Selector Setting
0 0	0M
0 1	M
1 0	2M
1 1	3M

To achieve 3M, we could add 2M to A (S), shift straight into S (A), add M to S (A) and shift right into A (S). As an alternative, we could create a special adder between M and the M selector OR's such that 3M appears at its outputs shortly after M is loaded. With this device, we could simply select 3M, add it to A (S), and right-shift the result into S (A). Neither approach is satisfactory, since the first is too slow and the second is too expensive. A better solution is achieved by recoding the multiplier to take advantage of the fact that addition and subtraction are equally easy to implement in the two's complement system.

Let

$$y = -2y_{-1} + y_0 + \sum_{k=1}^{44} y_k 2^{-k} \quad ,$$

represent the rounded multiplier. Because of round-off, y may have a value

of +1 so y_{-1} is not always equal to y_0. In order to take advantage of the base 4 shift between A,Q and S,R, two bits of the multiplier are examined at each step. Therefore, it is convenient to express y in terms of 23 base 4 digits, β_i.

$$y = \beta = \beta_0 + \sum_{i=1}^{22} \beta_i 4^{-i}$$

where

$$\beta_0 = -1, 0, 1$$

and

$$\beta_i' = -1, 0, 1, 2$$

for

$$1 \leq i \leq 22$$

We require that $\beta = \beta'$. This can be guaranteed if we relate B_i' to B_i as follows:

$$\beta_i' = \beta_i + \lambda_i - 4\lambda_{i-1}$$

where

$$\lambda_i = 0, 1$$

for

$$1 \leq i \leq 21$$

with

$$\lambda_{-1} = 0$$

and

$$\lambda_{22} = 0$$

The λ_i are called mode bits.

To prove that $\beta' = \beta$ if the β_i' are related to the β_i as indicated above, we consider $\beta' - \beta$.

$$\beta' - \beta = \beta_0 + \lambda_0 + \sum_{i=1}^{22} (\beta_i + \lambda_i - 4\lambda_{i-1})4^{-i} - \beta_0 - \sum_{i=1}^{22} \beta_i 4^{-i}$$

$$= \lambda_0 + \sum_{i=1}^{22} \lambda_i 4^{-i} - \sum_{j=1}^{22} \lambda_{j-1} 4^{-(j-1)}$$

Let $j = i + 1$ in the second sum and remember that $\lambda_{22} = 0$.

$$\beta' - \beta = \lambda_0 + \sum_{i=1}^{21} \lambda_i 4^{-i} - \sum_{i=0}^{21} \lambda_i 4^{-i}$$

$$\beta' - \beta = \lambda_0 - \lambda_0 = 0$$

The rule for choosing λ_{i-1} given β_i and λ_i is stated below for $1 \leq i \leq 22$.

$$\lambda_{i-1} = 1 \quad \text{if} \quad 3 \leq \beta_i + \lambda_i \leq 4$$

$$\lambda_{i-1} = 0 \quad \text{if} \quad 0 \leq \beta_i + \lambda_i \leq 2$$

This is not the only rule that might be used to choose λ_{i-1} if 2M, M, 0M, $\overline{M}$, and $\overline{2M}$ are available options for both the MsA and MsS selectors. The choice of this rule was partially based on the M selector requirements of

the (B) and (D) sequences. Since a binary division algorithm is employed, the $\overline{2}$MsA option is not needed. However, all five of the above options for the MsS selector are used in the (B) sequence. By recoding the rounded multiplier according to the rule given above, it is possible to eliminate the $\overline{2}$MsA option entirely. Note that $-1 \leq \beta_i' \leq 2$ when the λ_i are chosen by this rule.

	β_i	λ_i	β_i'	Selector Setting	λ_{i-1}
Positive Mode	0	0	0	0M	0
	1	0	1	M	0
	2	0	2	2M	0
	3	0	-1	$\overline{M}$	1
Negative Mode	0	1	1	M	0
	1	1	2	2M	0
	2	1	-1	$\overline{M}$	1
	3	1	0	0M	1

At the time of recoding, β_i is located in R_{41}, R_{42} or Q_{41}, Q_{42} while the mode bit, λ_i, is in R^*_{42} or Q^*_{42}.

The Boolean expressions for the μMsA decoder (D-1506) can be derived from the above table. On the $(22-i)^{\underline{th}}$ step of the multiplication process, let $\beta_i = 2r_{41} + r_{42}$ and let $\lambda_i = r^*_{42}$. Negative logic is used to set the selector, since it responds to zero inputs. The signals M3 - μ_2MsA, M5 - a and M5 - b go to zero when the designated control points become active and thereby cause the μMsA decoder outputs to set the MsA selector. Let μ_2Aa = (M3 - μ_2Ms)(M5 - a) and μ_2Ab = (M3 - μ_2MsA)(M5 - b).

$$\overline{2}\text{MsA} = (\overline{r}_{41} \vee r_{42} \vee r^*_{42} \vee \mu_2\text{Ab})(r_{41} \vee \overline{r}_{42} \vee \overline{r}^*_{42} \vee \mu_2\text{Ab})$$

$$\text{MsA} = (r_{41} \vee \overline{r}_{42} \vee r^*_{42} \vee \mu_2\text{Ab})(r_{41} \vee r_{42} \vee \overline{r}^*_{42} \vee \mu_2\text{Ab})$$

$$\text{OMsA} = (r_{41} \vee r_{42} \vee r^*_{42} \vee \mu_2 Aa)(\bar{r}_{41} \vee \bar{r}_{42} \vee \bar{r}^*_{42} \vee \mu_2 Aa)$$

$$\bar{\text{M}}\text{sA} = (\bar{r}_{41} \vee \bar{r}_{42} \vee r^*_{42} \vee \mu_2 Aa)(\bar{r}_{41} \vee r_{42} \vee \bar{r}^*_{42} \vee \mu_2 Aa)$$

The AND in these equations is actually formed at the input to the MsA selector logic shown on MsA-I (D-1176).

The positive logic used to set Q^*_{42} is shown on LQR (D-1524). The signal, μ, is the true output of the μ status memory element which is set to "1" during M1 and reset to "0" during the last pass through M6. (See the Delayed Control Flow Chart.)

$$Q^{*\,\prime}_{42} = r_{41}r_{42} \vee r_{41}r^*_{42}$$

Note that $Q^{*\,\prime}_{42} = \lambda_i$ which is the mode bit associated with the $(i\text{-}1)^{\underline{\text{th}}}$ base 4 multiplier digit and the $(22 - i+1)^{\underline{\text{th}}}$ step of the multiplication.

Except for slight modifications, the logic shown above applies to the μMsS decoder as well. The modifications consist of substituting q for r and ANDing in another term to permit recoding the least-significant multiplier digit as α is transferred into R following round-off. Since the least-significant mode bit always equals zero, $\lambda_{22} = 0$, the values, of β'_{22} and λ'_{21} depend only on α_{43} and α_{44} as indicated below. In order to abbreviate the request signals, we let $\mu_0 S = M2 - \mu_0 \text{MsS}$, $\mu_1 Sb = (M4 - \mu_1 \text{MsS})(M6 - b)$ and $\mu_1 Sc = (M4 - \mu_1 \text{MsS})(M6 - c)$.

$$\text{2MsS} = (\bar{q}_{41} \vee q_{42} \vee q^*_{42} \vee \mu_1 Sb)(q_{41} \vee \bar{q}_{42} \vee \bar{q}^*_{42} \vee \mu_1 Sb)(\bar{\alpha}_{43} \vee \alpha_{44} \vee \mu_0 S)$$

$$\text{MsS} = (q_{41} \vee \bar{q}_{42} \vee q^*_{42} \vee \mu_1 Sc)(q_{41} \vee q_{42} \vee \bar{q}^*_{42} \vee \mu_1 Sb)(\alpha_{43} \vee \bar{\alpha}_{44} \vee \mu_0 S)$$

$$\text{OMsS} = (q_{41} \vee q_{42} \vee q^*_{42} \vee \mu_1 Sb)(\bar{q}_{41} \vee \bar{q}_{42} \vee \bar{q}^*_{42} \vee \mu_1 Sb)(\alpha_{43} \vee \alpha_{44} \vee \mu_0 S)$$

$$\overline{MsS} = (\overline{q}_{41} \vee \overline{q}_{42} \vee q^*_{42} \vee \mu_1 Sc)(\overline{q}_{41} \vee q_{42} \vee \overline{q^*_{42}} \vee \mu_1 Sc)(\overline{\alpha}_{43} \vee \alpha_{44} \vee \mu_0 S)$$

As before, the AND in these equations is actually formed at the input to the MsS selector logic shown on MsS-I (D-1178).

The positive logic used to set R^*_{42} is shown on LQR (D-1524). Since KgA = 1 as the rounded multiplier, α, is being transferred to R, it is used to condition the setting of R^*_{42} on the initial step. During the iterative loop of multiplication (control points M5 and M6) KgA = 0, so R^*_{42} is set according to the contents of Q_{41}, Q_{42}, and Q^*_{42}.

$$R^{*\,\prime}_{42} = q_{41}q_{42}(\overline{KgA}) \vee q_{41}q^*_{42}(\overline{KgA}) \vee \alpha_{43}\alpha_{44}(KgA)$$

It is important to note the role of the μ status memory element (D-1299). Its output controls the inputs to Q^*_{42} and R^*_{42} as indicated by the above equations. Q^*_{42} and R^*_{42} cannot be set to 1 by the multiplier recoding logic if $\mu = 0$, i.e., if $\overline{\mu}$ is true. During the last pass through M6, the count in the EAU reaches -1 and $\overline{\mu}$ is set as indicated by the Delayed Control Flow Chart. This means that R^*_{42} is set to zero during the last right shift into S,R. The two sign bits of the multiplier (the sign digit) are in Q_{41},Q_{42} at the start of this step. During the step, they are recoded as a regular bit pair and the MsS selector is set accordingly. However, if R^*_{42} is set to "1" as a consequence of this recoding, trouble arises on the following step. The borrow-subtractor subtracts a unit from R_{41},R_{42} if $R^*_{42} = 1$. Thus, if $R^*_{42} = 1$, the two least significant bits of the product which are in R_{41},R_{42} at the end of M6 on the final pass would be reduced by a unit during the straight transfer into Q_{41},Q_{42} which occurs in M7. The need to set $R^*_{42} = 0$ during the last pass through M6 is clear.

A comment is also in order regarding the way in which the "sign digit" of the rounded multiplier is recoded. This digit is represented by the sign bits of the rounded multiplier, α_{-1} and α_0. Since the multiplier is normalized or nearly normalized prior to rounding, the bit pair, $\alpha_{-1}\alpha_0$ may be 11, 00, or 01, but never 10. Furthermore, 01 can only arise when the rounded multiplier is

+1 = 01.00 ... 00. It follows that β_0 may be -1, 0, or +1. However, at the time β_0 is recoded, it is in Q_{41}, Q_{42}, and Q^*_{42} may be 0 or 1 if β_0 = -1 or 0; but Q^*_{42} = 0 if β_0 = +1. The recoding takes place during the final pass through M6, so β_0' determines the setting of the MsS selector.

A case analysis follows. It was stated earlier that λ_{-1} = 0, so $\beta_0' = \beta_0 + \lambda_0$.

If β_0 = -1, the $Q_{41} = Q_{42} = 1$. The μMsS decoder logic sees β_0 as +3 instead of -1. If $\lambda_0 = Q^*_{42} = 0$, β_0' = -1 and λ_{-1} = 1. $\overline{\text{MsS}}$ is set, but λ_{-1}, which is placed in R^*_{42}, is forced to zero because μ is set to "0" during this step. Consequently, β_0' = -1 as it should and λ_{-1} = 0. If $\lambda_0 = Q^*_{42} = 1$, β_0' = 0 as it should so OMsS is set. Since μ is set to "0", λ_{-1} is again forced to zero even though it would have unit value otherwise.

If β_0 = 0, then $Q_{41} = Q_{42} = 0$. The μMsS decoder logic sees β_0 as a zero and sets OMsS if λ_0 = 0, and MsS if λ_0 = 1. In both cases, λ_{-1} would be a zero anyway so the setting of μ to "0" is not essential.

If β_0 = +1, then Q_{41} = 0 and Q_{42} = 1. Since this can only arise when the rounded multiplier is +1, all previous multiplier digits must be zero. The equations for R^*_{41} and Q^*_{42} show that λ_0 will always be zero in this case. In fact, $R^*_{42} = Q^*_{42} = 0$ at every step of the multiplication. Therefore, the μMsS decoder interprets β_0' as +1 and sets MsS. Here again λ_{-1} = 0, so the setting of μ to "0" is not essential.

3.1.16 The Division Predictors (ρ Decoders)

The ρMsA and -MsS division predictor logic is shown on DPα (D-1507). It is used only during division. The outputs of this logic serve two functions. First, they set the MsA and MsS selector mechanisms so that the appropriate multiples of the divisor are subtracted from the shifted partial remainder at each step of the binary division process. Secondly, they are fed to the

quotient digit recoder logic, shown at the top of DPα (D-1507), which produces the base 4 quotient digit that is inserted in R^*_{42}, R_{43}, and R_{44} on every second step of the division process.

To provide a background for the discussion which follows, a brief description of the division process is included at this point. A complete description is given in section 5.13.

In the MAU a modified, binary nonrestoring division algorithm is used to divide the subnormalized fractional dividend, $\frac{1}{16} \leq |D'| \leq \frac{1}{4}$, the "normalized" fractional divisor, $\frac{1}{4} \leq |d| \leq 1$. The recursive relationships are given below. The rules for selecting the quotient "bit" (y_{2j} or y_{2j+1} = -1, 0, or 1) are complicated due to the stored carry representation of partial remainders. Consequently, the discussion of ρMsA and ρMsS logic (i.e., the selection rules) is relegated to the latter half of this section.

$$s_{2j+1} = 4(a_{2j} - y_{2j}d)$$

$$a_{2j+2} = s_{2j+1} - 2y_{2j+1}d$$

$$j = 0, 1, 2 \ldots 22 \text{ or } 23$$

The number of steps in the division process is controlled to yield a "normalized" fractional quotient, $\frac{1}{4} \leq |q| \leq 1$.

a_0 = the subnormalized fractional dividend, $\frac{1}{16} \leq |D'| \leq \frac{1}{4}$, in AQ with $j = -1$.

d = the "normalized" fractional divisor, $\frac{1}{4} \leq |d| \leq 1$.

s_{2j+1} = the $(2j+1)^{\underline{th}}$ shifted partial remainder in SR.

a_{2j+2} = the $(2j+2)^{\underline{th}}$ shifted partial remainder in AQ.

y_{2j} = the quotient "bit" (y_{2j} = 1, 0, or -1) generated by the ρMsA logic during the straight transfer in AQ.

y_{2j+1} = the quotient "bit" (y_{2j+1} = 1, 0, or -1) generated by the ρMsS logic during the left shift into SR.

In all three divide orders DIV, NDV, and VID, the divisor and dividend are normalized prior to the fractional division. This is accomplished by the (R) sequence and the first four steps of the (D) sequence. During D5, the exponent difference is calculated. If the dividend is nonzero, it is subnormalized in D6, i.e., shifted right one base 4 position into S,R. This is done to insure that the fractional quotient is always in the range: $-1 \leq f \leq 1$.

The modified, binary nonrestoring division process begins at this point. During D7, the MsS selector is set to 0MsS. The subnormalized dividend is gated straight down into A,Q and the ρMsA predictor logic, which examines the most significant σ outputs of the S adder, sets the MsA selector to MsA, 0MsA, or $\overline{\text{M}}$sA.

During D9, the most significant half of the first partial remainder is formed at the output of the A adder and shifted left into S. The least significant half of this partial remainder is in Q and is shifted left into S and R as usual. Bits Q_{-1} and Q_0 go into S_{43} and S_{44} as indicated in section 3.1.2.8. During this transfer, the ρMsS predictor logic examines the most significant bits of the α output and sets the MsS selector to 2MsS, 0MsS, or $\overline{2\text{M}}$sS. At the same time, the quotient "bit" recoder logic (section 3.1.17) recodes the zero quotient "bit" that is associated with the 0MsS setting during D7 and the quotient "bit" (y_0 = 1, 0, -1) that is associated with the ρMsA setting during D9. The resulting base 4 quotient digit (1, 0, -1) is gated into R_{43}, R_{44} (01, 00, 11) during D9. R^*_{42} is always set to zero at this point to prevent the borrow-subtractor (section 3.1.17) from modifying the two least significant bits of the dividend which are placed in R_{41} and R_{42} during D9. If R^*_{42} = 1 during the first pass through D10, the borrow-subtractor reduces this bit configuration by a unit in the $42^{\underline{\text{nd}}}$ position.

After D9 is complete, the control passes to the iterative loop of

After D9 is complete, the control passes to the iterative loop of division which consists of control points D10 and D11. In D10 the new partial remainder which appears at the output of the S adder is gated straight into A,Q. The ρMsA predictor sets the MsA selector to MsA, OMsA, or $\overline{\text{M}}$sA according to the most significant bits of this partial remainder. The associated quotient "bit" (y_{2j} = 1, 0, -1) is stored in G_2, H_2; and the quotient digit that was predicted and stored during D9 or D11 is transferred from G_1',H_1' to G_1,H_1. In D11, the new partial remainder which appears at the output of the A adder is shifted left one base 4 position and gated into S,R. The ρMsS predictor sets the MsS selector to 2MsS, OMsS, or $\overline{2\text{M}}$sS according to the most significant bits of this partial remainder. The corresponding quotient "bit" (y_{2j+1} = 1, 0, -1) is stored in G_1',H_1'. The quotient "bits" that were predicted on the two previous steps are held in G_1,H_1 and G_2,H_2 during D11. The quotient bit recoder examines the output of these memory elements and produces a base 4 quotient digit in the range of -3 to +3 which is inserted in R^*_{42}, R_{43}, and R_{44} during this control step.

This process is repeated until the count in the EAU becomes negative (indicating that 23 base 4 quotient digits have been generated) and the generated quotient appears normalized in R. If the latter condition is not met, 24 base 4 quotient digits are generated. (See the description of the R normalization detector in section 3.1.10.) When the quotient appears normalized in R, control passes to D12 where the final partial remainder and quotient are gated straight into A and Q respectively and the MsA selector is set to KgA. The final steps of the (D) sequence round the quotient, generate a true remainder, and compute the associated exponents.

Having briefly described the function of the division predictors, we proceed to define their logic. Note that negative logic is used to set

the selectors as usual. For example, $\overline{MsA} - \rho$ must be zero to set the MsA selector to $\overline{MsA}$.

ρMsA Predictor Logic

$$\xi_2 = m_{-1}(\overline{s\sigma_{-2}}) \vee \overline{m_{-1}}(s\sigma_{-2})$$

$$\overline{\xi}_2 = m_{-1}(s\sigma_{-2}) \vee \overline{m}_{-1}(\overline{s\sigma_{-2}})$$

The signals $s\sigma_{-2}$ and $\overline{s\sigma_{-2}}$ are outputs of the high S special adder as discussed in section 3.1.4.

$$\eta_2 = \overset{*}{\sigma}_{-2} \vee \sigma_{-1}\sigma_0\overset{*}{\sigma}_0$$

$$\overline{\eta}_2 = \overline{\overset{*}{\sigma}_{-2}}\ \overline{\sigma_{-1}} \vee \overline{\overset{*}{\sigma}_{-2}}\ \overline{\sigma_0} \vee \overline{\overset{*}{\sigma}_{-2}}\ \overline{\overset{*}{\sigma}_0}$$

$$MsA - \rho = \xi_2\eta_2 \vee \overline{\xi}_2\overline{\eta}_2 \vee \overline{OMsA} - \rho$$

$$\overline{MsA} - \rho = \xi_2\overline{\eta}_2 \vee \overline{\xi}_2\eta_2 \vee \overline{OMsA} - \rho$$

$$OMsA - \rho = \overline{\sigma_{-1}} \vee \overline{\sigma_0} \vee \overline{\sigma_1} \vee \overline{\sigma_2}\rho MsA$$

$$\overline{OMsA} - \rho = \sigma_{-1}\sigma_0\sigma_1\sigma_2 \vee \rho MsA$$

$$\rho MsA = (D7 - \rho_1 MsA)(D10 - a)$$

In order for the ρMsA predictor logic to set the MsA selector, ρMsA must go to zero as the result of a request from control point D7 or D10. (See section 4.2.) In the absence of such requests, the outputs have unit value and therefore do not affect the MsA selector.

The G_2 and H_2 Eccles-Jordan memory elements are used to hold the predicted quotient "bit" which is interpreted as +1, 0, or -1. If $MsA - \rho = 0$, then the divisor is <u>added</u> to the partial remainder on the next step and the corresponding quotient "bit is -1. If $\overline{MsA} - \rho = 0$,

then the divisor is subtracted from the partial remainder and the corresponding quotient "bit" is +1. If $\bar{M}sA - \rho = 0$, the corresponding quotient "bit" is zero. A zero input on the 1 or 0 side of an Eccles-Jordan causes the output on that side to assume unit value. The significance of the true outputs of G_2 and H_2 in terms of the quotient "bit" stored is summarized below.

Quotient "Bit" Stored = y_{2j} (j = 0, 1, 2 --- 22 or 23)	g_2	h_2
+1	0	0
0	1	1
0	1	1
-1	1	0

Let the 1 and 0 side inputs to G_2 and H_2 be designated as $G_2, \bar{G}_2$ and $H_2 \bar{H}_2$ respectively.

$$G_2 = (MsA - \rho)(OMsA - \rho) \qquad \bar{G}_2 = \bar{M}sA - \rho$$

$$H_2 = OMsA - \rho \qquad \bar{H}_2 = \overline{OMsA} - \rho$$

The AND at the input to G_2 insures that $g_2 = 1$ whenever $OMsA - \rho = 0$.

ρMsS Predictor Logic

$$\xi_1 = m_{-1}\bar{\alpha}_{-1} \vee \bar{m}_{-1}\alpha_{-1}$$

$$\bar{\xi}_1 = m_{-1}\alpha_{-1} \vee \bar{m}_{-1}\alpha_{-1}$$

$$\eta_1 = \alpha_0\alpha_0^* \vee \alpha_0\alpha_1\alpha_2\alpha_2^*$$

$$\bar{\eta}_1 = \bar{\alpha}_0 \vee \bar{\alpha}_0^*\bar{\alpha}_1 \vee \bar{\alpha}_0^*\bar{\alpha}_2 \vee \bar{\alpha}_0^*\bar{\alpha}_2^*$$

$$2MsS - \rho = \xi_1\eta_1 \vee \bar{\xi}_1\bar{\eta}_1 \vee \overline{0MsS} - \rho$$

$$\overline{2MsS} - \rho = \xi_1\bar{\eta}_1 \vee \bar{\xi}_1\eta_1 \vee \overline{0MsS} - \rho$$

$$0MsS - \rho = \bar{\alpha}_0 \vee \bar{\alpha}_1 \vee \bar{\alpha}_2 \vee \bar{\alpha}_3 \vee \rho MsS$$

$$\overline{0MsS} - \rho = \alpha_0\alpha_1\alpha_2\alpha_3 \vee \rho MsS$$

$$\rho MsS = (D9 - \rho_1 MsS)(D11 - a)$$

The ρMsS signal must go to zero as the result of a request from D9 or D11 in order for the ρMsS logic to set the MsS selector. As long as $\rho MsS = 1$, the outputs of the ρMsS predictor have unit value and therefore do not affect the MsS selector.

The G_1' and H_1' Eccles-Jordan memory elements are set as follows:

$$G_1' = (2MsS - \rho)(0MsS - \rho) \qquad \bar{G}_1' = \overline{2MsS} - \rho$$

$$H_1' = (0MsS - \rho)(gH_1') \qquad \bar{H}_1' = \overline{0MsS} - \rho$$

The gH_1' signal goes to zero during D6 to insure that $h_1' = 1$ at the time H_1 is set in D7. As a result, $h_1 = 1$ when the quotient "bit" recoder produces the initial base 4 quotient digit during D9. (See section 3.1.17.)

Quotient "Bit" Stored = y_{2j+1} (j = 0, 1, 2 --- 22 or 23)	g_1'	h_1'
+1	0	0
0	p	1
0	p	1
-1	1	0

During D7, p = 0 or 1 depending on how g_1 was left at the end of the previous divide order. During any pass through D10 in which $h_1' = 1$, p = 1 because of the AND at the input to G_1'.

The logic for gating the contents of G_1', H_1' into G_1, H_1 is given below.

$$G_1 = \overline{g_1'} \vee \rho MsA \qquad \overline{G}_1 = g_1' \vee \rho MsA$$

$$H_1 = \overline{h_1'} \vee \rho MsA \qquad \overline{H}_1 = h_1' \vee \rho MsA$$

The quotient "bit" ($y_{2j+1} = 1, 0, -1$) that is predicted during the present pass through D9 or D11 is not used in forming the base 4 quotient digit that is inserted in R^*_{42}, R_{43}, and R_{44} during this pass. Instead, it is held in G_1', H_1' and transferred to G_1, H_1 during D10. In D7 or D10 the quotient "bit" ($y_{2j} = 1, 0, -1$) predicted by the ρMsA logic is stored in G_2, H_2. During the following pass through D9 or D11 the two quotient "bits" stored in G_1, H_1, G_2, and H_2 are recoded as a base 4 quotient digit which is inserted in R^*_{42}, R_{43} and R_{44}. Note that y_{2j+1} always has twice the weight of y_{2j}. The quotient bit recoder is described in the next section.

Zero quotient "bits" arise as a consequence of the stored carry representation of partial remainders. The true sign of fraction, f, in stored carry representation is not always determined by its leading bits. If the number of leading bits that one is willing to examine is fixed, there is a range of f, $-t \leq f \leq u$, for which these bits may or may not determine the true sign. If f is outisde this range, the true sign is always determined by these bits. The number of leading bits that must be examined increases as t and u approach zero. If the unshifted partial remainder, f, appearing at the output of the A or S adders is sufficiently large, its true sign can be determined by the associated predictor. If this sign agrees (disagrees) with the sign of the divisor,

m_{-1}, the divisor or twice the divisor is subtracted from (added to) the shifted partial remainder on the next step; and a positive (negative) quotient "bit" is generated. If $-t \leq f \leq u$, the predictor may or may not be able to determine the true sign. Consequently, the sign comparison may be incorrect. In this case, zero is subtracted from (added to) the shifted partial remainder, giving rise to a zero quotient "bit".

The last half of this section is devoted to a study of the underlying basis of the ρMsA and ρMsS logic. The recursive relations are used to determine the limits on the partial quotients and partial remainders. Knowledge of the latter is fundamental to understanding the design of the ρMsA and ρMsS logic.

Dividing the recursive relations by d, we obtain the following expressions for successive partial quotients:

$$Q_{2j+1} = \frac{s_{2j+1}}{d} = 4Q_{2j} - 4y_{2j}$$

$$Q_{2j+2} = \frac{a_{2j+2}}{d} = Q_{2j+1} - 2y_{2j+1}$$

where y_{2j} and y_{2j+1} = -1, 0, or 1 and j = 0, 1, 2 --- 22 or 23.

To find the steady-state positive limits on the partial quotients, let $y_{2j} = y_{2j+1} = 1$ and let $(Q_{2j+2})_{Max} = (Q_{2j})_{Max}$. Substituting the second expression into the first, we obtain

$$(Q_{2j+1})_{Max} = 4[(Q_{2j+1})_{Max} - 2] - 4$$

It follows that $(Q_{2j+1})_{Max} = 4$ and $(Q_{2j+2})_{Max} = 2$.

The steady-state negative limits are obtained in the same way with $y_{2j} = y_{2j+1} = -1$ and $(Q_{2j+2})_{Min} = (Q_{2j})_{Min}$.

$$(Q_{2j+1})_{Min} = 4[(Q_{2j+1})_{Min} + 2] + 4$$

Therefore, $(Q_{2j+1})_{Min} = -4$ and $(Q_{2j+2})_{Min} = -2$.

Figures 4 and 5 provide a graphic picture of the generation of successive partial quotients. The partial quotients have no particular significance except that they are not a function of d, and therefore Figs. 4 and 5 are valid for all d, $\frac{1}{4} \leq |d| \leq 1$. Furthermore, the limits on the partial remainders can be easily derived from the limits on the partial quotients.

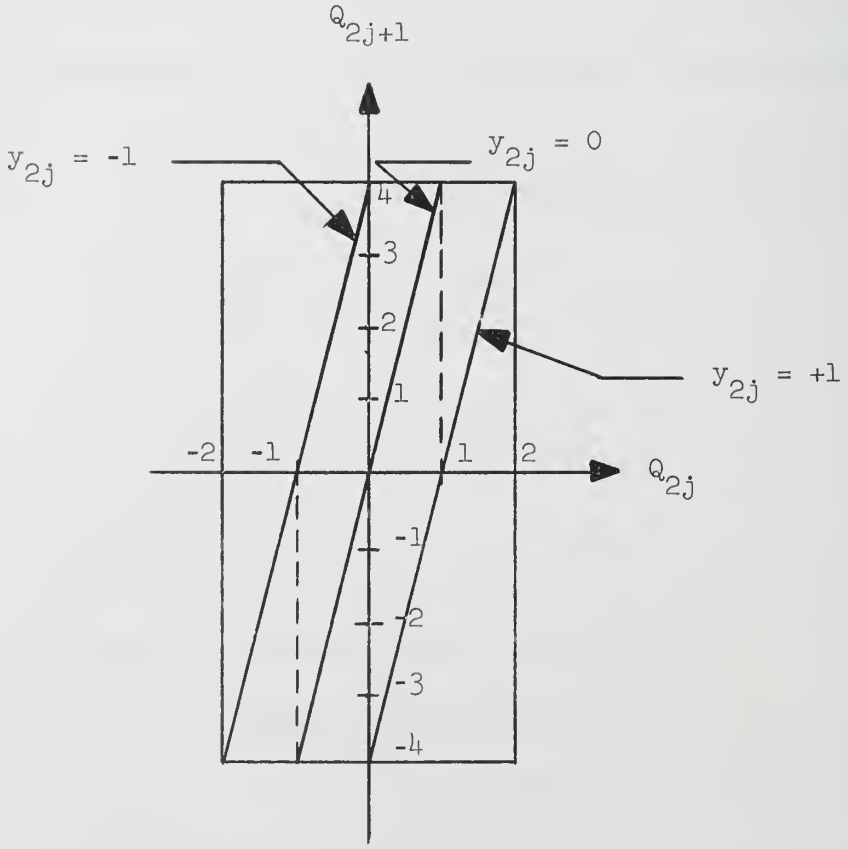


Figure 4. $Q_{2j+1} = 4Q_{2j} - 4y_{2j}$

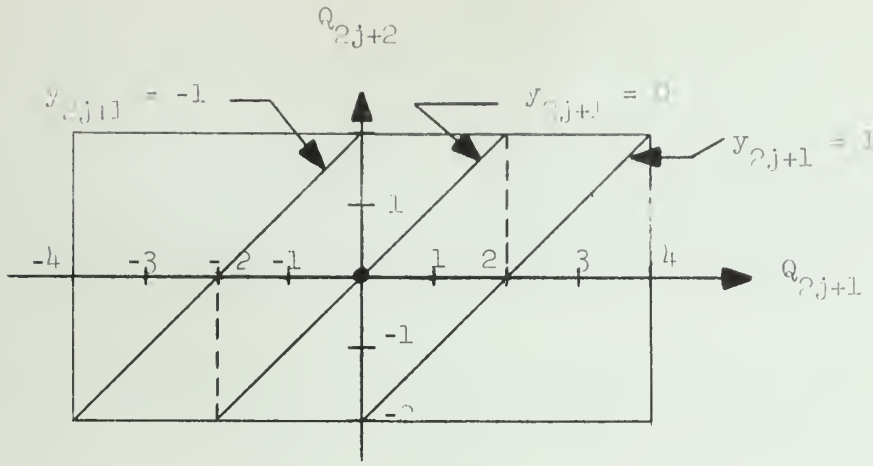


Figure 5. $Q_{2j+2} = Q_{2j+1} - 2y_{2j+1}$

In Fig. 4, note that $y_{2j} = 0$ is not allowed unless $-1 \leq Q_{2j} \leq 1$. Likewise, in Fig. 5, $y_{2j+1} = 0$ is not allowed unless $-2 \leq Q_{2j+1} \leq 2$. These limits may be translated into limits on the unshifted partial remainders σ_{2j} and α_{2j+1}. Using the expressions for Q_{2j+2} and Q_{2j}, the limits on Q_{2j} imply $-|d|_{Min} \leq a_{2j} \leq |d|_{Min}$, and the limits on Q_{2j+1} imply $-2|d|_{Min} \leq s_{2j+1} \leq 2|d|_{Min}$. The limits on $|d|$ are $\frac{1}{4} \leq |d| \leq 1$ where both $-\frac{1}{4}$ and $+1$ may arise as the result of round-off in a VID order. (See section 5.13.) Since $\sigma_{2j} = a_{2j}$ and $\alpha_{2j+1} = \frac{s_{2j+1}}{4}$, σ_{2j} must lie in the range $-\frac{1}{4} \leq \sigma_{2j} \leq \frac{1}{4}$ before $y_{2j} = 0$ is allowed, and α_{2j+1} must lie in the range $-\frac{1}{8} \leq \alpha_{2j+1} \leq \frac{1}{8}$ before $y_{2j+1} = 0$ is allowed.

The outer limits on the unshifted partial remainders σ_{2j} and α_{2j+1} are determined in a similar manner. Since $-2 \leq Q_{2j} \leq 2$, $-2|d|_{Max} \leq a_{2j} \leq 2|d|_{Max}$ which implies $-2 \leq \sigma_{2j} \leq 2$. Likewise, $-4 \leq Q_{2j+1} \leq 4$ implies $-1 \leq \alpha_{2j+1} \leq 1$.

We are now in a position to consider the number of leading bits of σ_{2j} and α_{2j+1} that the ρMsA and ρMsS logic must examine to predict the appropriate

quotient "bit". There are two problems here. First, we determine the number of "bits" to the right of the point that are necessary to insure that σ_{2j} or α_{2j+1} is within the prescribed limits when a zero quotient "bit" is predicted. Secondly, we decide how many "bits" to the left of the point are needed to determine accurately the sign when the unshifted partial remainders assume their maximum values.

To insure that the true sign of σ is known whenever $\sigma < -\frac{1}{4}$ or $\sigma > \frac{1}{4}$, $s\sigma_{-2}$, $\overset{*}{\sigma}_{-2}$, σ_{-1}, σ_0, and $\overset{*}{\sigma}_0$ are examined. The basis for this assertion is discussed later. The apparent sign of σ is $s\sigma_{-2} \oplus \eta_2$. η_2 is the apparent carry into the -2 position, and is defined by one of the equations of the ρMsA logic. If $s\sigma_{-2} \oplus \eta_2 = 0$, σ must be positive since any stored carries to the right of $\overset{*}{\sigma}_0$ have positive weight. If $s\sigma_{-2} \oplus \eta_2 = 1$, σ is apparently negative but may, in fact, be positive due to stored carries to the right of $\overset{*}{\sigma}_0$. Therefore, the negative inner limit, $\sigma = -\frac{1}{4}$, determines the number of leading bits of σ which must be examined to insure that a zero quotient "bit" is predicted if and only if $-\frac{1}{4} \leq \sigma \leq \frac{1}{4}$.

Assuming the divide control sequence is functioning properly, $\sigma_{-1}\sigma_0\sigma_1\sigma_2 = 1$ implies that $s\sigma_{-2} = 1$. If $s\sigma_{-2} = 0$ under this condition, σ would be greater than +2 which is outside the limits of the division process as discussed above. Furthermore, $\sigma_{-1}\sigma_0\sigma_1\sigma_2 = 1$ together with the adder relation guarantees that $\overset{*}{\sigma}_{-2} = \overset{*}{\sigma}_0 = 0$. Consequently, if $\overset{*}{\sigma}_2 = 1$, the true sign of σ is positive while the apparent sign is negative. This also applies for any stored carry to the right of $\overset{*}{\sigma}_2$, provided that there is a string of units between it and σ_2. The condition $\sigma_{-1}\sigma_0\sigma_1\sigma_2 = 1$ also guarantees that $-\frac{1}{4} \leq \sigma < \frac{1}{6}$. The positive limit is the sum of the infinite stored carry representation shown below.

0 0 1 1 1 1

1 1 1 1.1 1 0 1 0 1 . . . 0 1 . . .

If $\sigma_{-1}\sigma_0\sigma_1\sigma_2 = 0$, a stored carry in the σ_2^* position or to the right of the σ_2^* position cannot propagate into the -2 position and affect the sign of σ. The apparent sign must therefore agree with the true sign. Note that $\sigma_{-1}\sigma_0\sigma_1\sigma_2 = 0$ does not imply that $\sigma < -\frac{1}{4}$ or $\frac{1}{6} < \sigma$. Thus, there are stored carry representations of α in the range $-\frac{1}{4} \geq \sigma < \frac{1}{6}$ for which the apparent sign must agree with the true sign.

The function of the ρMsA logic is summarized as follows. The apparent carry into the -2 position is $\eta_2 = \sigma_{-2}^* \vee \sigma_{-1}\sigma_0\sigma_0^*$. The apparent sign of σ, $s\sigma_{-2} \oplus \eta_2$, is compared with the true sign of the divisor, m_{-1}. In negative logic, this comparison is defined by $\overline{\text{MsA}} - \rho = \xi_2 \oplus \eta_2$ where $\xi_2 = m_{-1} \oplus s\sigma_{-2}$. This comparison is meaningful if and only if the apparent sign of α and its true sign agree. If $\sigma_{-1}\sigma_0\sigma_1\sigma_2 = 0$, this agreement is guaranteed, so $\overline{\text{MsA}}$ is selected if $\overline{\text{MsA}} - \rho = 0$ (i.e., if the signs of the dividend and divisor agree), and MsA is selected if MsA - ρ = 0 (i.e., if the signs of the dividend and divisor disagree). If $\sigma_{-1}\sigma_0\sigma_1\sigma_2 = 1$, the apparent sign of σ may not agree with the true sign of σ, so OMsA is selected. This selection does not cause the division process to go out of bounds in S (i.e., $s_{2j+1} = 4\alpha_{2j+1}$ within $-4|d| \leq s_{2j+1} \leq 4|d|$). Assuming the division process is functioning properly, this is a direct consequence of the fact that $-\frac{1}{4} \leq \sigma < \frac{1}{6}$ when $\sigma_{-1}\sigma_0\sigma_1\sigma_2 = 1$.

The discussion of the limits on $\alpha = \alpha_{2j+1}$ follows a similar pattern. The bits α_{-1}, α_0, α_0^*, α_1, α_2, α_2^*, and α_3 are examined to insure that the true sign of α is know whenever $\alpha < -\frac{1}{8}$ or $\alpha > \frac{1}{8}$. This discussed in more detail later. The apparent sign of α is $\alpha_{-1} \oplus \eta_1$ where $\eta_1 = \alpha_0\alpha_0^* \vee \alpha_0\alpha_1\alpha_2\alpha_2^*$ is the apparent carry into the -1 position. If $\alpha_{-1} \oplus \eta_1 = 0$, α must be positive. If $\alpha_{-1} \oplus \eta_1 = 1$, α is apparently negative, but the true sign may be positive due

to stored carries to the right of α_2^*. It follows that the negative inner limit, $\alpha = -\frac{1}{8}$, determines the number of leading bits which must be examined to insure that a zero quotient "bit" is predicted if and only if $-\frac{1}{8} \leq \alpha \leq \frac{1}{8}$.

If $\alpha_0\alpha_1\alpha_2\alpha_3 = 1$, then $\alpha_{-1} = 1$, assuming the division process is functioning properly (i.e., assuming the limits on the partial remainders have been maintained). If $\alpha_{-1} = 0$ when $\alpha_0\alpha_1\alpha_2\alpha_3 = 1$, α must be greater than +1 and the limits on the division process have been exceeded (i.e., an error of some type has occurred). Note that when $\alpha_0\alpha_1\alpha_2\alpha_3 = 1$, the adder relation guarantees that $\alpha_0^* = \alpha_2^* = 0$. Therefore, under this condition the apparent sign of α is negative, but the true sign is positive if a string of units and a stored carry appear to the right of α_3. As a final note, $\alpha_0\alpha_1\alpha_2\alpha_3 = 1$ insures that $-\frac{1}{8} \leq \alpha < \frac{1}{24}$. The positive limit is the sum of the infinite stored carry representation shown below.

$$\begin{array}{cccccccccc} & 0 & & 0 & & 1 & & 1 & & 1 \\ 1 & 1.1 & 1 & 1 & 1 & 0 & 1 & \ldots & 0 & 1 \ldots \end{array}$$

If $\alpha_0\alpha_1\alpha_2\alpha_3 = 0$, the apparent sign of α must agree with its true sign. Any stored carry to the right of α_3 cannot be propagated into the -1 position. Note that α may be in the range $-\frac{1}{8} \leq \alpha < \frac{1}{24}$ even though $\alpha_0\alpha_1\alpha_2\alpha_3 = 0$.

The function of the ρMsS logic is summarized as follows. The apparent carry into the -1 position, $\eta_1 = \alpha_0\alpha_0^* \vee \alpha_0\alpha_1\alpha_2\alpha_2^*$, determines the apparent sign of α, $\alpha_{-1} \oplus \eta_1$. The latter is compared with the true sign of the divisor, m_{-1}. This comparison is formed as $\overline{2}$MsS - $\rho = \xi_1 \oplus \eta_1$ in negative logic where $\xi_1 = \alpha_{-1} \oplus m_{-1}$. The result of this comparison is valid if and only if the true and apparent signs of α agree. If $\alpha_0\alpha_1\alpha_2\alpha_3 = 0$, agreement is guaranteed, so $\overline{2}$MsS is selected if $\overline{2}$MsS - $\rho = 0$ (i.e., if the signs of the dividend and divisor agree), and 2MsS is selected if 2MsS - $\rho = 0$ (i.e., if the signs of the dividend

and divisor disagree). If $\alpha_0\alpha_1\alpha_2\alpha_3 = 1$, OMsS is selected, since the apparent and true signs of α may not agree. Even though OMsS is selected, the succeeding partial remainder $\sigma_{2j+2} = a_{2j+2}$ is still within $-2|d| \leq a_{2j+2} \leq 2|d|$. If the division process is functioning properly, this result follows directly from the fact that $-\frac{1}{8} \leq \alpha < \frac{1}{24}$ when $\alpha_0\alpha_1\alpha_2\alpha_3 = 1$.

The maximum and mimimum values which the partial remainders may attain is the last topic considered in this section. These values are used to determine the number of bits to the left of the point that must be examined to extablish the true sign of σ or α in all cases.

It has been shown that under steady-state conditions, the limits on the unshifted partial remainders are $-2 \leq \sigma \leq 2$ and $-1 \leq \alpha \leq 1$. However, this does not guarantee that the limiting values can actually arise in practice. It is shown below that more realistic limits on the unshifted partial remainders are $-2 \leq \sigma < \frac{5}{3}$ and $-1 \leq \alpha < \frac{11}{12}$.

Replacing a_{2j+2} by σ_{2j+2} and s_{2j+1} by $4\alpha_{2j+1}$ in the recursive relations, we obtain

$$\alpha_{2j+1} = \sigma_{2j} - y_{2j}d$$

$$\sigma_{2j+2} = 4\alpha_{2j+1} - 2y_{2j+1}d$$

Keeping the ρMsA and ρMsS logic in mind, we can now determine the maximum and minimum values of α_{2j+1} and σ_{2j+2}.

The limits on α_{2j+1} are determined by noting that σ_{2j} and $y_{2j}d$ have the same sign if $\sigma_{-1}\sigma_0\sigma_1\sigma_2 = 0$ and may or may not have the same sign if $\alpha_{-1}\sigma_0\sigma_1\sigma_2 = 1$. However, in the latter case, we know that $y_{2j} = 0$. It is clear that if $\sigma_{2j} = 0$ and $d = |1|$, then $y_{2j}d = 1$ and $\alpha_{2j+1} = -1$. In order for $-y_{2j}d = +1$, $|d|$ must be 1 and σ_{2j} must be negative with $\sigma_{-1}\sigma_0\sigma_1\sigma_2 = 0$. To make

α_{2j+1} as positive as possible, σ_{2j} must be the smallest negative value for which $\sigma_{-1}\sigma_0\sigma_1\sigma_2 = 0$. This value is found to be $-\frac{1}{12}$. It cannot occur in the machine because it has the infinite representation shown below.

$$\sigma_{2j} = 1\ 1\ 1.1\ 0\ 0\ 1\ 0\ 1\ \overset{0}{\ldots}\ 0\ 1\ \overset{1}{\ldots} = -\frac{1}{12}$$

Therefore, if $|d| = 1$, the most positive α_{2j+1} must be less than $1 - \frac{1}{12} = \frac{11}{12}$. The limits on α_{2j+1} are: $-1 \leq \alpha_{2j+1} < \frac{11}{12}$, and the corresponding limits on $s_{2j+1} = 4\alpha_{2j+1}$ are: $-4 \leq s_{2j+1} < \frac{11}{13}$.

To obtain the limits on σ_{2j+2}, we note that α_{2j+1} and $y_{2j+1}d$ have the same sign if $\alpha_0\alpha_1\alpha_2\alpha_3 = 0$ and may or may not have the same sign otherwise. If $\alpha_0\alpha_1\alpha_2\alpha_3 = 1$, $y_{2j+1} = 0$. The negative limit on σ_{2j+2} occurs when $\alpha_{2j+1} = 0$ and $|d| = 1$, since $\sigma_{2j+2} = 4(0) - 2 = -2$. The positive limit occurs when $|d| = 1$ and α_{2j+1} is the smallest negative value for which $\alpha_0\alpha_1\alpha_2\alpha_3 = 0$. This value is $-\frac{1}{12}$ and has the infinite representation shown below.

$$\alpha_{2j+1} = 1\ 1.1\ 1\ 0\ 1\ 0\ 1\ \overset{0}{\ldots}\ 0\ 1\ \overset{1}{\ldots} = -\frac{1}{12}$$

If $|d| = 1$, the most positive σ_{2j+2} is less that $4(-\frac{1}{12}) + 2 = \frac{5}{3}$. The limits on $\sigma_{2j+2} = a_{2j+2}$ are: $-2 \leq \sigma_{2j+2} < \frac{5}{3}$.

It is interesting to note that $\alpha_{2j+1} = \frac{11}{12}$ and $\sigma_{2j+2} = \frac{5}{3}$ are not steady-state limits even if infinite representations were allowed. The limit $\sigma_{2j+2} = \frac{5}{3}$ can be obtained assuming $\alpha_{2j+1} = \frac{11}{12}$, but not conversely.

The following examples illustrate how the limits on the unshifted partial remainders arise or are approached.

Let the subnormalized dividend be $\frac{1}{8}$ and the normalized divisor be -1.

						0		0		0					
j = -1	AQ				0	0	.	0	1	0	. . .	=		$\frac{1}{8}$	
j = 0	+ M				1	1	.	0	0	0	. . .	=	-	1	$y_0 = -1$
						0		0		0					
j = 0	α				1	1	.	0	1	0	. . .	=	-	$\frac{7}{8}$	
				0		0		0		0					
j = 0	SR	1	1	0	0	.	1	0	0	0	. . .	=	-	$\frac{7}{2}$	
j = 0	-2M	0	0	1	0	.	0	0	0	0	. . .	=	+	2	$y_1 = 1$
				0		0		0		0					
j = 0	σ			1	1	0	.	1	0	0	0	. . .	=	-	$\frac{3}{2}$
								0		0					
j = 0	AQ				1	0	.	1	0	0	0	. . .	=	-	$\frac{3}{2}$
j = 1	- M				0	1	.	0	0	0	0	. . .	=	+	1 $y_2 = 1$
						0		0		0					
j = 1	α				1	1	.	1	0	0	0	. . .	=	-	$\frac{1}{2}$
				0		0		0		0					
j = 1	SR	1	1	1	0	.	0	0	0	0	. . .	=	-	2	
j = 1	-2M	0	0	1	0	.	0	0	0	0	. . .	=	+	2	$y_3 = 1$
				0		0		0		0					
j = 1	σ			0	0	0	.	0	0	0	0	. . .	=		0
						0		0		0					
j = 1	AQ				0	0	.	0	0	0	0	. . .	=		0
j = 2	M				1	1	.	0	0	0	0	. . .	=	-	1 $y_4 = -1$
						0		0		0					
j = 2	α				1	1	.	0	0	0	0	. . .	=	-	1
				0		0		0		0					
j = 2	SR	1	1	0	0	,	0	0	0	0	. . .	=	-	4	
j = 2	-2M	0	0	1	0	.	0	0	0	0	. . .	=	+	2	$y_5 = 1$
				0		0		0		0					
j = 2	σ			1	1	0	.	0	0	0	0	. . .	=	-	2
						0		0		0					
j = 2	AQ				1	0	.	0	0	0	0	. . .	=	-	2
j = 3	- M				0	1	.	0	0	0	0	. . .	=	+	1 $y_6 = 1$
						0		0		0					
j = 3	α				1	1	.	0	0	0	0	. . .	=	-	1

The negative limits of $\sigma = -2$ and $\alpha = -1$ clearly occur in this example. It is easily seen that the infinite quotient is

$$y = -1.1\ 1\ 1\ -1\ 1\ 1\ 1\ 1\ \ldots 1\ \ldots = -\frac{1}{8}$$

Let the subnormalized dividend by $-\frac{1}{16} - 2^{-88}$ and let the normalized divisor be -1.

						0		0		0		0					
j = -1	AQ				1	1	.	1	1	1	0	1	1	. . .	=	$-\frac{1}{16} - 2^{-88}$	
j = 0	OM				0	0	.	0	0	0	0	0	0	. . .	=	0	$y_0 = 0$
						0		0		0		0					
j = 0	α				1	1	.	1	1	1	0	1	1	. . .	=	$-\frac{1}{16} - 2^{-88}$	
				0		0		0		0							
j = 0	SR	1	1	1	1	.	1	0	1	1	.	.	.	. . .	=	$-\frac{1}{4} - 2^{-86}$	
j = 0	OM	0	0	0	0	.	0	0	0	0	.	.	.	. . .	=	0	$y_1 = 0$
				0		0		0		0							
j = 0	σ			1	1	1	.	1	0	1	1	.	.	. . .	=	$-\frac{1}{4} - 2^{-86}$	
						0		0		0							
j = 0	AQ				1	1	.	1	0	1	1	.	.	. . .	=	$-\frac{1}{4} - 2^{-86}$	
j = 1	- M				0	1	.	0	0	0	0	.	.	. . .	=	+ 1	$y_2 = 1$
						0		0		0							
j = 1	α				0	0	.	1	0	1	1	.	.	. . .	=	$+\frac{3}{4} - 2^{-86}$	
				0		0		0		0							
j = 1	SR	0	0	1	0	.	1	1	1	1	.	.	.	. . .	=	$+3 - 2^{-84}$	
j = 1	2M	1	1	1	0	.	0	0	0	0	.	.	.	. . .	=	- 2	$y_3 = -1$
				0		0		0		0							
j = 1	σ		0	0	0	.	1	1	1	1	.	.	.	. . .	=	$1 - 2^{-84}$	
						0		0		0							
j = 1	AQ				0	0	.	1	1	1	1	.	.	. . .	=	$1 - 2^{-84}$	
j = 2	M				1	1	.	0	0	0	0	.	.	. . .	=	- 1	$y_4 = -1$
						0		0		0							
j = 2	α				1	1	.	1	1	1	1	.	.	. . .	=	-2^{-84}	

Although the last partial remainder shown in this example has a small negative value, the ρMsS and ρMsA logic will predict zero quotient "bits" for

almost all of the remaining steps of the division process because $\alpha_0\alpha_1\alpha_2\alpha_3 = 1$ and $\sigma_{-1}\sigma_0\sigma_1\sigma_2 = 1$. It is clear that $y \tilde{=} 0.01\text{-}1\text{-}1000\ldots = +\frac{1}{16}$. In this example the largest value of σ is approximately +1, while the largest value of α is approximately $+\frac{3}{4}$. Other examples using divisors slightly less than unity in absolute value will yield unshifted partial remainders which are closer to the limits of $\sigma = \frac{5}{3}$ and $\alpha = \frac{11}{12}$. There are no examples which yield unshifted partial remainders that equal or exceed these limits.

Let $\sigma k_{-1} = \sigma_0 c_0^*$ represent the apparent carry into the σ_{-1} position. Assuming $\sigma_{-1}\sigma_0\sigma_1\sigma_2 = 0$, what are the smallest positive and negative values of σ for which $\sigma_{-1} \oplus \sigma k_{-1}$ does <u>not</u> represent the true sign of σ ? The smallest positive value is +2. It may assume either of the following forms under the above conditions:

$$\begin{array}{cccccccccc} & 0 & & 1 & & 0 & & 0 & & \\ 0 & 0 & 0 & 1 & . & 0 & 0 & 0 & 0 & \ldots; \end{array} \qquad \begin{array}{cccccccccc} & & & & & 0 & & 0 & & 0 & & 0 \\ 0 & 0 & 1 & 0 & . & 0 & 0 & 0 & 0 & \ldots \end{array}$$

The smallest negative value under these conditions is $-1\frac{5}{6}$. It has the following infinite representation:

$$\begin{array}{cccccccccccc} & 0 & & 0 & & 1 & & 1 & & 1 & & 1 \\ 1 & 1 & 0 & 1 & . & 1 & 1 & 0 & 1 & 0 & 1 \ldots & 0\ 1 \ldots \end{array}$$

Although this value cannot arise as an unshifted partial remainder, -2 certainly can. It may assume a number of different representations for which $\sigma_{-1} \oplus \sigma k_{-1}$ does not yield the true sign of σ . Two possible representations are given below.

$$\begin{array}{cccccccccc} & 0 & & 0 & & 1 & & 0 & & \\ 1 & 1 & 0 & 1 & . & 1 & 1 & 0 & 0 & \ldots; \end{array} \qquad \begin{array}{cccccccccc} & & & & & 0 & & 0 & & 0 & & 1 & & 0 \\ 1 & 1 & 0 & 1 & . & 1 & 1 & 1 & 1 & 0 & 0 & \ldots \end{array}$$

It is clear from this analysis that when $\sigma_{-1}\sigma_0\sigma_1\sigma_2 = 0$, $\sigma_{-1} \oplus \sigma k_{-1}$ always yields the true sign of the unshifted partial remainder, σ , when $\sigma > 0$

but fails to yield the true sign in certain cases when $\sigma < 0$. To avoid this, it is necessary to examine $s\sigma_{-2} \oplus \eta_2$ where $\eta_2 = \sigma^*_{-2} \vee \sigma_{-1}\sigma_0\sigma^*_0$. If $\sigma_{-1}\sigma_0\sigma_1\sigma_2 = 0$, the smallest positive and negative values of σ for which $s\sigma_{-1} \oplus \eta_2$ does not represent the true sign are $+4$ and $-3\frac{5}{6}$. It follows that $s\sigma_{-2} \oplus \eta_2$ represents the true sign of σ when $\sigma_{-1}\sigma_0\sigma_1\sigma_2 = 0$ and $-2 \leq \sigma < \frac{5}{3}$.

Let $\alpha k_0 = \alpha^*_0 \vee \alpha_0\alpha_1\alpha_2\alpha^*_2$ represent the apparent carry into the α_0 position. Assuming $\alpha_0\alpha_1\alpha_2\alpha_3 = 0$, the smallest positive and negative values of α for which $\alpha_0 \oplus \alpha k_0$ does not represent the true sign of α are $+1$ and $-\frac{23}{24}$. Since the unshifted partial remainder, α, may be -1, $\alpha_0 \oplus \alpha k_0$ will not yield the true sign of α in all cases when $\alpha_0\alpha_1\alpha_2\alpha_3 = 0$. To avoid this difficulty, it is necessary to examine $\alpha_{-1} \oplus \eta_1$ where $\eta_1 = \alpha_0\alpha^*_0 \vee \alpha_0\alpha_1\alpha_2\alpha^*_2$. Assuming $\alpha_0\alpha_1\alpha_2\alpha_3 = 0$, the smallest positive and negative values of α for which $\alpha_{-1} \oplus \eta_1$ does not represent the true sign are $+2$ and $-1\frac{23}{24}$. Therefore, $\alpha_{-1} \oplus \eta_1$ represents the true sign of α when $\alpha_0\alpha_1\alpha_2\alpha_3 = 0$ and $-1 \leq \alpha < \frac{5}{3}$.

In the original design of the ρMsA and ρMsS logic, $\sigma_{-1} \oplus \sigma k_{-1}$ and $\alpha_0 \oplus \alpha k_0$ were thought to represent the true signs of σ and α when $\sigma_{-1}\sigma_0\sigma_1\sigma_2 = 0$ $\alpha_0\alpha_1\alpha_2\alpha_3 = 0$ respectively. This error was pointed out by R. H. Farrell and corrected before layout was complete.

3.1.17 The Quotient "Bit" Recoder and the Borrow-Subtractor

The quotient "bit" recoder logic is shown at the top of DPα (D-1507). It is used only during division to recode successive quotient "bits", which are held in G_1, H_1, G_2, and H_2 into base 4 quotient digits. At the output of the recoder, the quotient digit is represented by $q_j = -4\beta_{42} + 2\rho_{43} + \rho_{44}$ where $j = 0, 1, 2 - - - 22$ or 23 and $-3 \leq q_j \leq 3$. (The q_j does not refer to the output of the Q_j F-element.) The bits β_{42}, ρ_{43}, and ρ_{44} are transferred to R^*_{42}, R_{43}, and R_{44} respectively when $gR = 1$, $\delta_1 = 1$, and $\delta_2 = 1$, indicating that a divide order is being executed. (See D-1524, D-1272, and D-1295.)

In the table shown below, y_{2j+1} is the quotient "bit" held by G_1,H_1; y_{2j+2} is the quotient "bit" held by G_2,H_2. The sign comparison signal, γ, is generated by the logic discussed in section 3.1.11. $\gamma = 1$ when the true signs of A,Q (e.g., the shifted partial remainder) and M (e.g., the divisor) disagree.

$2y_{2j+1}$	y_{2j+2}	γ	q_j	β_{42}	ρ_{43}	ρ_{44}
-2	-1	e	-3	1	0	1
-2	0	e	-2	1	1	0
-2	1	e	-1	1	1	1
0	-1	e	-1	1	1	1
0	0	1	0	1	0	0
0	0	0	0	0	0	0
0	1	e	1	0	0	1
2	-1	e	1	0	0	1
2	0	e	2	0	1	0
2	1	e	3	0	1	1

The e symbol denotes either a "1" or a "0" , and j = 0, 1, 2 --- 22 or 23.

From this table we obtain the following expressions:

$$\beta_{42} = (2y_{2j+1} = -2) \vee (2y_{2j+1} = 0)(y_{2j+2} = -1) \vee (2y_{2j+1} = 0)(y_{2j+2} = 0)(\gamma = 1)$$

Using the Boolean expressions for y_{2j+1} and y_{2j+2} as given in the preceding section, we obtain:

$$\beta_{42} = g_1\bar{h}_1 \vee h_1g_2\bar{h}_2 \vee h_1h_2\gamma$$

$$\rho_{43} = (2y_{2j+1} = -2)\Big[(y_{2j+2} = 0) \vee (y_{2j+2} = 1)\Big] \vee (2y_{2j+1} = 0)(y_{2j+2} = -1) \vee (2y_{2j+1} = 2)\Big[(y_{2j+2} = 0) \vee (y_{2j+2} = 1)\Big]$$

it follows that

$$\rho_{43} = g_1\bar{h}_1\left[h_2 \vee \bar{g}_2\bar{h}_2\right] \vee h_1 g_2\bar{h}_2 \vee \bar{g}_1\bar{h}_1\left[h_2 \vee \bar{g}_2\bar{h}_2\right]$$

$$= \bar{h}_1 h_2 \vee \bar{h}_1\bar{g}_2 \vee h_1 g_2\bar{h}_2$$

$$\rho_{44} = (y_{2j+2} = -1) \vee (y_{2j+2} = 1)$$

This yields

$$\rho_{44} = \bar{h}_2$$

In the preceding section, it is noted $h_2 = 1$ implies $g_2 = 1$, and $h_1 = 1$ implies $g_1 = 1$ except during the D9 control step. Therefore, we can say $\bar{g}_2 h_2 \equiv 0$ during D9 and D11, and $\bar{g}_1 h_1 \equiv 0$ during D11. Examination of the above expressions for β_{42}, ρ_{43}, and ρ_{44} shows these identities would not produce any simplification of the recoder logic even if the hardware necessary to insure $\bar{g}_1 h_1 \equiv 0$ during D9 were added.

The borrow-subtractor logic appears on the upper half of SEC (D-1527). It is used only during division to convert serially the base 4 representation of the quotient ($-3 \leq q_j \leq 3$) to the usual binary representation. If the quotient "bit" recoder produces a negative base 4 quotient digit, R^*_{42} is set to "1" when gR = 1. The borrow-subtractor propagates this borrow over the R_{42} and R_{41} positions. The outputs of the borrow-subtractor are defined in the table shown below.

r_{41}	r_{42}	r^*_{42}	d_{41}	d_{42}
0	0	0	0	0
0	0	1	1	1
0	1	0	0	1
0	1	1	0	0
1	0	0	1	0
1	0	1	0	1
1	1	0	1	1
1	1	1	1	0

The corresponding Boolean expressions for d_{41} and d_{42} follow:

$$d_{41} = r_{41}r_{42} \vee r_{41}\bar{r}_{42} \vee \bar{r}_{41}\bar{r}_{42}r^*_{42}$$

$$d_{42} = r_{42}\bar{r}^*_{42} \vee \bar{r}_{42}r^*_{42}$$

The second line of the table shows a unit subtracted from zero in R_{41} and R_{42} yields three as a result (i.e., $d_{41} = d_{42} = 1$). This is meaningful if there is a unit in the R_{40} position before the subtraction and a zero afterwards. Theoretically, this is precisely what happens even though the borrow-subtractor does not examine an output in the R_{40} position. For the present, we shall assume the existence of an "R^*_{40}" F-element which always contains a unit when $r_{41} = r_{42} = 0$ and $r^*_{42} = 1$, but contains a zero otherwise.

The quotient "bit" recoder, borrow-subtractor, and $Q^{*\,"}_{42} = r^*_{42}\bar{r}_{43}\bar{r}_{44}$ (see sections 3.1.2.11 and D-1524) operate as a unit to form the fractional quotient in binary representation in the R and Q registers during control steps D7, D9, D10, and D11. The function of this logic during the formation of a fractional quotient is now examined.

In control step D6 (see D-1128 or D-1272) the normalized dividend in A,Q is shifted right into S,R placing the two least significant bits in R_{43}

and R_{44}. At this point in the division sequence, R^*_{42} and Q^*_{42} both contain "0" and $\delta_1 = \delta_2 = 0$. Since the order is division and not multiplication, $\mu = 0$. Hence, the inputs to R^*_{42} and Q^*_{42} as defined in sections 3.1.2.13 and 3.1.2.11, are both zero. OMsS is selected, which implies the quotient "bit" $y_{-1} = 0$. This is a consequence of subnormalizing the dividend to obtain a fractional quotient in the range $-1 \leq q \leq 1$. The signal gH'_1 goes to "0" in D6 to set H'_1 to "1" which agrees with the selection of OMsS and $y_{-1} = 0$. The most significant half of the initial unshifted partial remainder appears at the output of the S-adder. The most significant bits of this output feed the ρMsA logic.

In D7, the subnormalized dividend or initial partial remainder is gated straight into A,Q. The two least significant bits go into Q_{43} and Q_{44}. The ρ_1MsA signal goes to "0", which permits the ρMsA logic to select $\overline{\text{M}}$sA, OMsA, or MsA and set G_2 and H_2 accordingly. It also permits the contents of G'_1 and H'_1 to be gated into G_1 and H_1. Following these operations, the output of G_2 and H_2 represents the y_0 quotient "bit" which may be 1, 0, or -1. The output of G_1 and H_1 represents the y_{-1} quotient "bit" which is zero, since $h_1 = 0$. Note that g_1 may be "1" or "0" depending on the contents of G'_1 during the final pass through D11 in the last divide operation. The most significant half of the first unshifted partial remainder appears at the output of the A-adder. The most significant bits of this output feed the ρMsS logic. The quotient "bit" recoder generates the initial base 4 quotient digit, $q_0 = +1$, 0, or -1, according to the outputs of G_1, H_1, G_2, and H_2.

Since $h_1 = 1$, q_0 and y_0 agree. If $y_0 = -1$, $\beta_{42} = 1$, $\rho_{43} = 1$, and $\rho_{44} = 1$. If $y_0 = 1$, $\beta_{42} = 0$, $\rho_{43} = 0$, and $\rho_{44} = 1$. If $y_0 = 0$, $\beta_{42} = \gamma$, $\rho_{43} = 0$, and $\rho_{44} = 0$. $\beta_{42} = \gamma = 1$ if the true sign of the subnormalized dividend in A,Q disagrees with the sign of the divisor. Thus, if $\sigma_{-1}\sigma_0\sigma_1\sigma_2 = 1$ and the true sign of σ disagrees with m_{-1}, a zero quotient "bit" is predicted ($y_0 = 0$), but a negative quotient bit will eventually arise. However, the value of β_{42} is not

important on this initial step. It is not gated into R^*_{42} during D9 because $\delta_2 = 0$ as explained below.

The first unshifted partial remainder is shifted left into S,R during D9. The two least significant bits are gated into R_{41} and R_{42}. The δ_1 signal goes to "1" allowing ρ_{43} and ρ_{44} to be gated into R_{43} and R_{44} respectively. The logic associated with the δ_1 signal appears in the D9 section of D-1272. The point at which this signal controls the inputs to R_{43} and R_{44} is shown in the right center section of D-1524. The Δ_2 Eccles-Jordan is still set to "0" (i.e., $\delta_2 = 0$), so a "0" instead of β_{42} is gated into R^*_{42}. The reason for this is given in the next paragraph. The ρ_1MsS signal also goes to "0" in D9, permitting the ρMsS logic to select $\overline{2}$MsS, 0MsS, or 2MsS and set G'_1 and H'_1 accordingly. Since the output of the quotient "bit" recoder is still influenced by the contents of G_1 and H_1 at this point, the need for G'_1 and H'_1 is apparent. The most significant bits of the second unshifted partial remainder appear at the S-adder output and feed the ρMsA logic.

Control now passes to the iterative loop of the division sequence (D10 and D11). We consider the first pass through the loop.

When D10 is activated the first time, the second unshifted partial remainder is gated straight into A,Q. Since R^*_{42} was set to "0" during D9, the outputs of the borrow-subtractor, d_{41} and d_{42}, simply reflect the two least significant bits of the partial remainder which are gated into Q_{41} and Q_{42}. Likewise, the first two bits of the quotient in R_{43} and R_{44} are gated into Q_{43} and Q_{44}. If the first base 4 quotient digit is -1, both R_{43} and R_{44} contain a "1". Unless these bits are modified by the borrow-subtractor during the first pass through D11, they will become the negative sign bits of the fractional quotient. Similar statements apply for $q_0 = 0$ and +1. Q^*_{42} is set to "0" since $R^*_{42} = 0$ and $\mu = 0$. If R^*_{42} were set to "1" during D9, the two least significant bits

of the first partial remainder would be modified by the borrow-subtractor during the first pass through D10. This is why Δ_2 is not set to "1" until control passes to D10. The ρ_2MsA signal goes to "0" during D10 and allows the ρMsA logic to select $\overline{\text{M}}$sA, 0MsA, or MsA and set G_2 and H_2 accordingly. It also gates the contents G_1' and H_1' into G_1 and H_1. The third unshifted partial remainder then appears at the output of the A-adder and in the first 44 bits of Q.

During the first pass through D11, the third unshifted partial remainder is shifted left into S,R placing the two least significant bits in R_{39} and R_{40}. The unmodified sign bits of the quotient which correspond to q_0 are gated into R_{41} and R_{42}. The outputs of the quotient "bit" recoder are gated into R_{42}^*, R_{43}, and R_{44} since $\delta_1 = \delta_2 = 1$. A "0" is gated into R_{42}^* unless the base 4 quotient digit q_1 is negative or zero with $\gamma = 1$. The ρ_2MsS signal goes to "0" which allows the ρMsS logic to select $\overline{2\text{M}}$sS, 0MsS, or 2MsS and set G_1' and H_1' accordingly. The fourth unshifted partial remainder appears at the output of the S-adder and in the first 42 bits of R.

When D10 is activated the second time, the fourth unshifted partial remainder is shifted straight into A,Q with the least significant bits being placed in Q_{39} and Q_{40}. The bit configuration in Q_{41}, Q_{42}, Q_{42}^*, Q_{43}, and Q_{44} is considered below. The ρ_2MsA signal goes to "0" and allows the ρMsA logic to select $\overline{\text{M}}$sA, 0MsA, or MsA and set G_2 and H_2 accordingly. This also transfers the contents of G_1' and H_1' to G_1 and H_1. The fifth unshifted partial remainder then appears at the output of the A-adder and in the first 42 bits of Q.

If R_{42}^* contains a "0" during the second pass through D10, the first four bits of the quotient in R_{41}, R_{42}, R_{43}, and R_{44} are gated straight into Q_{41}, Q_{42}, Q_{43}, and Q_{44}. A "0" is gated into Q_{42}^*.

If R_{42}^* contains a "1" and either or both of R_{43} and R_{44} contain a "1", the base 4 quotient digit q_1 is negative. The outputs of the borrow-subtractor

(d_{41}, d_{42}) are (0,0), (1,1), or (1,0) depending on whether $q_0 = 1$, 0, or -1. If $q_0 = 0$, R_{41} and R_{42} both contain "0" and the borrow is false. However, in this case the outputs of the borrow-subtractor (1,1) are gated into Q_{41} and Q_{42} as the sign bits of the negative quotient. Since either or both of R_{43} and R_{44} contain a "1", a "0" is gated into Q^*_{42}. The contents of R_{43} and R_{44} are gated straight into Q_{43} and Q_{44} as usual.

If R^*_{42} contains a "1" and R_{43} and R_{44} both contain "0" during the second pass through D10, q_1 is zero but the true sign of the second partial remainder disagreed with the true sign of the divisor. Note that the second partial remainder is gated into A,Q during the first pass through D10. In this case, a borrow is initiated in anticipation of a future negative quotient digit. Even though a zero quotient digit is predicted, a negative quotient digit will eventually arise in the infinite representation of the quotient because $\gamma = 1$. Since the borrow which accompanies a negative quotient digit can only be propagated over R_{41} and R_{42} during the transfer from R to Q, it is necessary to initiate a borrow from the bits in R_{41} and R_{42} as soon as a zero quotient digit is placed in R_{43} and R_{44} with $\gamma = 1$. In view of the fact that 22 or 23 zero quotient digits may be generated before the negative quotient digit appears, the borrow from R_{41} and R_{42} must be held as a stored carry in Q^*_{42}. Therefore, if the division process terminates before the negative quotient digit is predicted, the stored carry is transferred to S^*_{42} by the interchange in D17 and added to the quotient by the S and A adders. Returning to the second pass through D10, the outputs of the borrow-subtractor are gated into Q_{41} and Q_{42}. $Q^{*\,"}_{42} = 1$ in this case, so a "1" is gated into Q^*_{42}. The zeros in R_{43} and R_{44} are gated straight into Q_{43} and Q_{44} as usual.

The significance of the theoretical "R^*_{40}" F-element is now apparent. If this element existed, it would receive the stored carry that is placed in Q^*_{42}

when a zero quotient digit is predicted with $\gamma = 1$. For example, assume the following bit configuration in R during the second pass through D10 where x and y represent the two least significant bits of the partial remainder.

$$"r^*_{40}" = 0 \qquad r^*_{42} = 1$$

$$r_{39} = x \quad r_{40} = y \quad r_{41} = 0 \quad r_{42} = 0 \quad r_{43} = 0 \quad r_{44} = 0$$

The resulting bit configuration in Q during the second pass through D11 is given below.

$$q^*_{42} = 1$$

$$q_{39} = x \quad q_{40} = y \quad q_{41} = 1 \quad q_{42} = 1 \quad q_{43} = 0 \quad q_{44} = 0$$

Assume that a zero quotient digit is placed in R_{43} and R_{44} during the second pass through D11. Since R^*_{42} was set to "1" under the same condition during the first pass through D11, γ must still be "1" so R^*_{42} is again set to "1". The resulting bit configurations in R and Q appear below.

$$"r^*_{40}" = 1 \qquad r^*_{42} = 1$$

$$r_{37} = x \quad r_{38} = y \quad r_{39} = 1 \quad r_{40} = 1 \quad r_{41} = 0 \quad r_{42} = 0 \quad r_{43} = 0 \quad r_{44} = 0$$

$$q^*_{42} = 1$$

$$q_{37} = x \quad q_{38} = y \quad q_{39} = 1 \quad q_{40} = 1 \quad q_{41} = 1 \quad q_{42} = 1 \quad q_{43} = 0 \quad q_{44} = 0$$

It is clear that the borrow in R^*_{42} is actually subtracted from $4"r^*_{40}" + 2r_{41} + r_{42}$. It is also clear that the "R^*_{40}" F-element does not have to exist physically. If we assume a -1 quotient digit is placed in R^*_{42}, R_{43}, and R_{44} during the second pass through D11, the resulting bit configurations in R and Q are as follows:

$$"r^{*}_{40}" = 1 \qquad r^{*}_{42} = 1$$

$$r_{37} = x \quad r_{38} = y \quad r_{39} = 1 \quad r_{40} = 1 \quad r_{41} = 0 \quad r_{42} = 0 \quad r_{43} = 1 \quad r_{44} = 1$$

$$q^{*}_{42} = 0$$

$$q_{37} = x \quad q_{38} = y \quad q_{39} = 1 \quad q_{40} = 1 \quad q_{41} = 1 \quad q_{42} = 1 \quad q_{43} = 1 \quad q_{44} = 1$$

This pattern of activity is typical of the remaining steps of the divide loop. During each pass through D11 a "1" is gated into R^{*}_{42} if the quotient "bit" recoder generates a negative base 4 quotient digit or a zero digit where the true signs of the partial remainder in A,Q and the divisor in M disagree (i.e., where $\gamma = 1$). In either case during the next pass through D10, a unit is borrowed from $4"r^{*}_{40}" + 2r_{41} + r_{42}$ to yield $2d_{41} + d_{42}$ which is gated into Q_{41} and Q_{42}. In the former case, the borrow is absorbed by the negative quotient digit that was placed in R^{*}_{42}, R_{43}, and R_{44} so Q^{*}_{42} is set to "0". In the latter case, a zero quotient digit was placed in R^{*}_{42}, R_{43}, and R_{44} so the borrow is held as stored carry in Q^{*}_{42}. If the quotient "bit" recoder logic generates a positive base 4 quotient digit or a zero digit with $\gamma = 0$ during any pass through D11, R^{*}_{42} is set to "0". A borrow from the bits in R_{41} and R_{42} does not occur during the next pass through D10, and Q^{*}_{42} is set to "0". Even if a string of zero quotient digits are generated, $\gamma = 0$ implies that the next nonzero quotient digit will be positive.

During the final pass through D11, the ρ_2MsS signal goes to "0" allowing the ρMsS logic to select $\overline{2M}$sS, 0MsS, or 2MsS and set G'_1 and H'_1 accordingly. The final base 4 quotient digit is placed in R^{*}_{42}, R_{43}, and R_{44}. Note that this digit is independent of the excess quotient "bit" held in G'_1 and H'_1. The divide loop terminates and control passes to D12.

The partial remainder beyond the final partial remainder appears at

the output of the S adder during D12. It is gated straight into A so its true sign may be compared with the sign of the divisor via the KgA request and the carry-borrow logic as explained in section 5.13. The output of the carry-borrow logic corresponds to γ at this point. It is gated into the CB F-element and used together with the excess quotient "bit" in G_1' and H_1' to determine quotient round-off as described in section 5.13. The latter occurs in D17 while the corresponding remainder correction occurs in D14.

The final unrounded fractional quotient appears in R during D12. If R^*_{42} contains a "1", the $41^{\underline{st}}$ and $42^{\underline{nd}}$ bits of the quotient are modified by the borrow-subtractor during the straight transfer into Q. Q^*_{42} is set to "1" if the last quotient digit is a zero and R^*_{42} is set to "1". This represents an unused borrow which is absorbed as a stored carry by the S and A adders following the interchange in D17.

This completes the description of the quotient "bit" recoder, borrow-subtractor and associated logic. Note on D-1272 that δ_1 goes to "0" during D12. Δ_2 is set to "0" during D14. For a description of the complete division control sequence the reader is referrec to section 5.13.

3.2 The Exponent Arithmetic Unit

The general structure of the EAU is shown in Fig. 6. The EA, ES, EM, and E registers each contain 8 bits as shown below.

$$EA_7, EA_6 \ldots EA_i \ldots EA_0$$
$$ES_7, ES_6 \ldots ES_i \ldots ES_0$$
$$EM_7, EM_6 \ldots EM_i \ldots EM_0$$
$$E_7, E_6 \ldots E_i \ldots E_0$$

The EAU does arithmetic modulo 256. The point lies at the right end

of all registers with the $i^{\underline{th}}$ bit having a weight of w^i except the $7^{\underline{th}}$ bit which has a weight of -2^7.

An 8 bit adder (D-adder) with an optional carry into the $0^{\underline{th}}$ position provides the capability of doing exponent arithemetic and counting. It accepts the outputs of the EA register and the sD selector as inputs, and yields a sum, d, which can be placed in ES via gES or in E via DgE. The value of d as function of em and ea under various selector and carry input settings is given below.

	c = 0	c = 1
EMsD:	d = ea + em	d = ea + em + 1
$\overline{\text{EM}}$sD:	d = ea - em - 1	d = ea - em
2sD:	d = ea + 2	d = ea + 3
0sD:	d = ea	d = ea + 1
- 2sD:	d = ea - 2	d = ea - 1
-22sD:	d = ea - 22	d = ea - 21

Note: The small c denotes the true output of the C status memory elements.

In contrast with the MsA and MsS selectors, the request for $\overline{\text{EM}}$sD does not automatically yield a carry input to the lowest order position such that d = ea - em. DC must also set C to "1" or "0" depending on whether d = ea - em or d = ea - em - 1 is desired at the adder output.

The selectors sEA and sE control the input to EA via gEA and the input to E via gE. The new contents of these registers following activation of the appropriate gate are listed below.

EsEA:	e $\xrightarrow{\text{gEA}}$ EA	EMsE:	em $\xrightarrow{\text{gE}}$ E
ESsEA:	es $\xrightarrow{\text{gEA}}$ EA	0sE:	0 $\xrightarrow{\text{gE}}$ E
0sEA:	0 $\xrightarrow{\text{gEA}}$ EA	6sE:	6 $\xrightarrow{\text{gE}}$ E
21sEA:	21 $\xrightarrow{\text{gEA}}$ EA	22sE:	22 $\xrightarrow{\text{gE}}$ E

During the decode step (G1) a zero or the contents of E is transferred to EM via FlgMEM. The sign bit of the incoming exponent IN_{45} is always gated

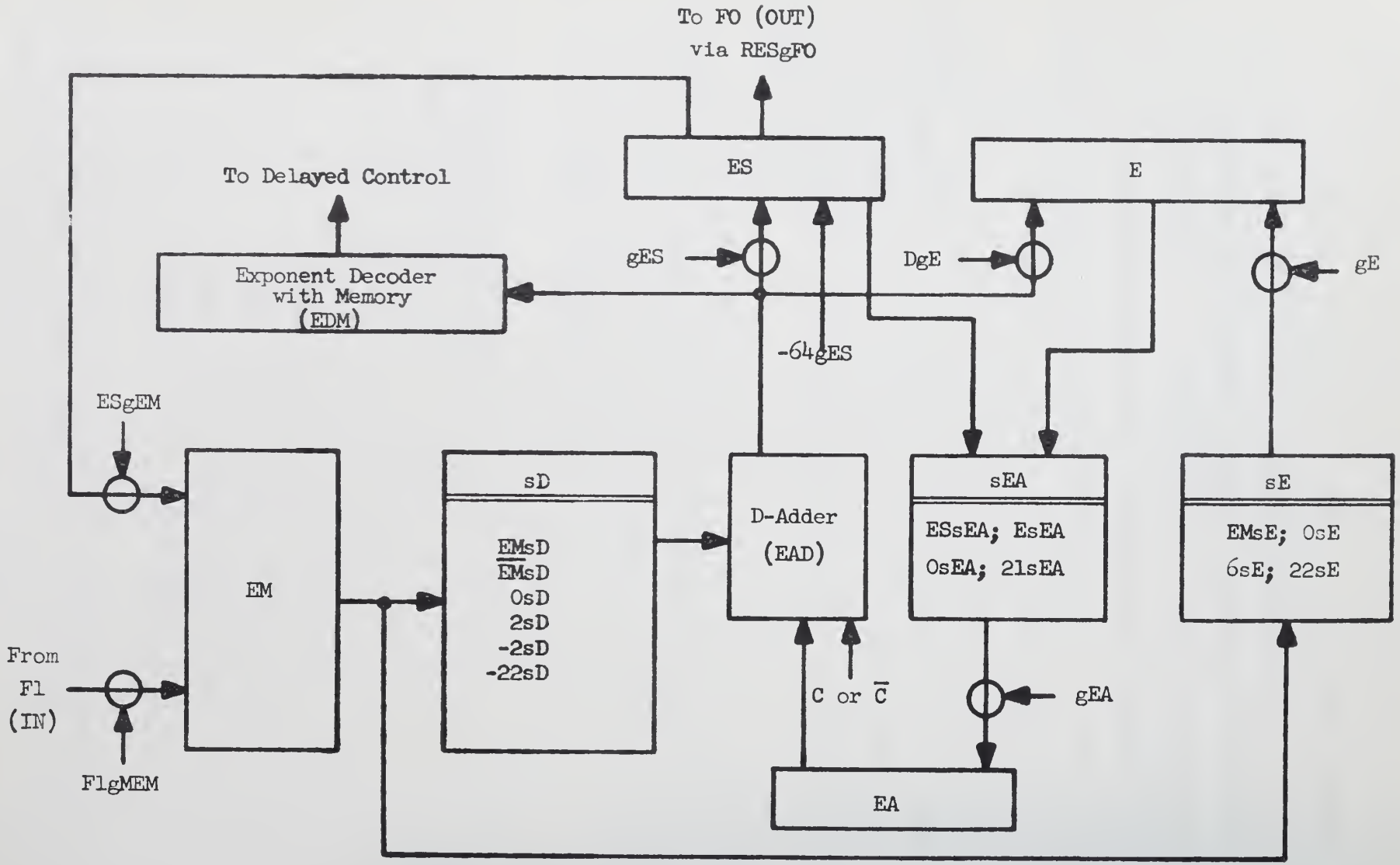


Figure 6. Block Diagram of the Exponent Arithmetic Unit

into both EM_6 and EM_7. The D adder is used to determine d = ea - em - 1 in floating addition, ea + em in multiplication, ea - em in division. It is also used in conjunction with the EA and ES or E registers for counting the number of shifts in floating addition and the number of iterative steps in multiplication and division. At the end of a DC operation the exponent of the result is left in E. At the end of divide orders the exponent of the quotient is in E and the exponent of the remainder is in ES.

In performing a DC store order, the exponent of the operand to be stored is placed in ES and then transferred to the last 7 bits of the OUT register during the S9 step of the store sequence (Flow Chart D-1128). Whereas $-128 \leq x < 128$ is allowed inside the EAU, only exponents in the range $-64 \leq x < 64$ can be stored with f. If x is stored alone as in SEX, es_7 is mapped into IN_{39}, IN_{40} ... IN_{43}, IN_{44} with $es_6 \longrightarrow IN_{45}$, $es_5 \longrightarrow IN_{46}$ ... and $es_0 \longrightarrow IN_{51}$. Using SEX, $-128 \leq x < 128$ can be stored as a quarter word by AC. Note that the sign bit is duplicated in the six most significant bit positions of this quarter word. The bits of f that normally occupy these positions are over-written during the transfer of R and ES to OUT. This feature is discussed in greater detail in section 3.1.2.14.

The strictly exponent orders complement the SEX order. Even though IN is copied into M and EM during the decode step as previously described, the first step of ADE, SDE, CAE, and CSE copies M_{44} into EM_7. Thus, if one is willing to use a 13 bit quarter word and ignore the fractional part or store it separately, storage and retrieval of full 8 bit exponents is possible.

The EAU decoder is a large block of logic whose inputs are the outputs of the D-adder. Its purpose is to detect specific values and ranges of the adder output. Knowledge of these values is used in the execution of the floating add and shift instructions. They are also used to determine the end of a

count. Detection of whether the output is inside or outside the range $-64 \leq x < 64$ is also accomplished at this point. Since knowledge of the previous range or value of d must be remembered during the time the inputs to the adder are changed, gES or DgE will gate the outputs of the EAU decoder into a register called ED. The memory elements of this register are named according to the range of d they represent. For example, the ES $\geq$ 0 memory element is set to the "1" state (i.e., set so that its 1 or true output is a positive voltage) by gES or DgE if $d \geq 0$ is true. If $d < 0$ is true when gES or DgE is on, the ES $\geq$ 0 memory element is set to the "0" state.

3.2.1 A Summary of the EAU Logic

The registers and selectors of the EAU are shown on the drawing by the same name (D-1502). The eight bit adder associated with the EAU is shown in block form on the EAU drawing and in detail on the EAD drawing (D-1503). The exponent decoder logic as shown on EDM (D-1504) is considered a part of the EAU. The outputs of the exponent adder feed the decoder as indicated by the central table on the EAU drawing.

3.2.2 The EAU Bit Path Logic

The bit path logic of the EAU is described first. As in the MAU, the description is register- and selector-oriented including the connections with the F1 (IN) and F0 (OUT) registers of the Flow Gate Memory. The descriptions of the 8-bit full carry adder and the exponent decoder follow in order.

3.2.2.1 The F1 to EM Path

The inputs to the EM register are shown at the top of the EAU drawing. When FlgMEM = 1, $F1_{45}$ is gated into EM_6 and EM_7 while F_{46} through F_{51} are gated

into EM_5 through EM_0 respectively. As shown in Fig. 6, there is a path from ES to EM. When ESgEM = 1, es_7 through es_0 are gated into EM_7 through EM_0 respectively. Note that m_{44} is gated into EM_7 when $m_{44}gEM_7 = 1$. This is shown in the upper left corner of D-1502. The first step of the (E) sequence is the only control step which causes this transfer. (See section 5.6.)

3.2.2.2 The EM Register and the EM = -64 Logic

The EM register is loaded as indicated in the preceding section. Its outputs feed the sD selector which is described in the following section and the sE selector which is discussed in section 3.2.2.7.

The true output of EM_6 and the complement outputs of EM_5 through EM_0 feed an AND shown in the top center of the EAU drawing.

$$(em = -64) = (em_6)(\overline{em_5})(\overline{em_4})(\overline{em_3})(\overline{em_2})(\overline{em_1})(\overline{em_0})$$

It is clear that the output of this AND has unit value only when EM contains ± 64. (The point in the EM and all other EAU registers is assumed to lie to the right of the $0^{\underline{th}}$ bit.) Since this detector is used by DC only when EM contains an in-range exponent from memory ($-64 \leq em < 64$), its output is called (em = -64).

3.2.2.3 The sD Selector

The outputs, eb_i, of the sD selector feed through difference amplifiers (shown at the bottom of the EAU drawing) to the 8-bit full-carry exponent adder (D adder). The eb_7 and $\overline{eb}_7$ signals also feed the exponent decoder. Shown below are the Boolean expressions for each bit position of the selector.

$$eb_i' = em_i(EMsD) \vee \overline{em}_i(\overline{EM}sD)$$

$$0 \leq i \leq 7$$

$$eb_7 = eb_7' \vee (-22sD) \vee (-2sD)$$

$$eb_6 = eb_6' \vee (-22sD) \vee (-2sD)$$

$$eb_5 = eb_5' \vee (-22sD) \vee (-2sD)$$

$$eb_4 = eb_4' \vee (-2sD)$$

$$eb_3 = eb_3' \vee (-22sD) \vee (-2sD)$$

$$eb_2 = eb_2' \vee (-2sD)$$

$$eb_1 = eb_1' \vee (-22sD) \vee (-2sD) \vee (2sD)$$

$$eb_0 = eb_0'$$

When the sD selector mechanism is set to OsD, EMsD = $\overline{\text{EM}}$sD = -2sD = 2sD = 0 In contrast with the $\overline{\text{M}}$sA, $\overline{\text{M}}$sS, and $\overline{\text{2M}}$sS settings in the MAU, $\overline{\text{EM}}$sD does not automatically cause the complement carry to be added to the least significant position of the difference. In the EAU adder (D adder) this carry is the true output, c, of the "complement carry" status memory element C (D-1285). If $\overline{\text{EM}}$sD and $\overline{\text{C}}$ are true during a particular control step, then ea - em - 1 appears at the D adder output. If $\overline{\text{EM}}$ and C are true, then ea - em appears at the output.

The D adder can be used to count up or down on the contents of EA by one or two units per pass. This is accomplished by a choosing of the proper combination of -2sD or 2sD and C or $\overline{\text{C}}$. For example, ea - 2 appears at the output of the D adder when -2sD is selected and $\overline{\text{C}}$ is true. This type of count is used to control the number of passes through the inner loop of the multiply sequence (M5 and M6).

The -22sD setting is used in only three control steps -- A4, S10, and D17. If EA contains the exponent of the accumulator, AQ, then ea - 22 is the exponent associated with the positive fraction contained by Q where the radix point lies to the left of q_{-i}.

3.2.2.4 The sEA Selector

As shown at the bottom of the EAU drawing, the sEA selector controls the input to the EA register. The Boolean expressions for the outputs, EA_i, are given below.

$$EA_i' = es_i(ESsEA) \vee e_i(EsEA)$$

$$0 \leq i \leq 7$$

The es_i and e_i signals are the true outputs of the ES_i and E_i F-elements respectively.

$$EA_7 = EA_7'$$

$$EA_6 = EA_6'$$

$$EA_5 = EA_5'$$

$$EA_4 = EA_4' \vee 2lsEA$$

$$EA_3 = EA_3'$$

$$EA_2 = EA_2' \vee 2lsEA$$

$$EA_1 = EA_1'$$

$$EA_0 = EA_0' \vee 2lsEA$$

When OsEA is selected, ESsEA = EsEA = 2lsEA = 0. The 2lsEA setting is used only in the M2 control step. The need for this additional selector option arose as a consequence of the way in which the count for the inner loop of multiply is handled and the desire to make the multiply sequence as fast as possible.

3.2.2.5 The EA Register

When gEA = 1, the output of the sEA selector is gated into the EA

register. The outputs of EA, ea_i, feed the exponent adder which is described in section 3.2.3. (Note on the EAU drawing that portions of this adder are on four chassis -- A6R, A7R, S6R, and S7R.) The ea_7 and $\overline{ea_7}$ signals also feed the exponent decoder.

3.2.2.6 The ES Register and the ES to F0 Path

As shown in Fig. 6, the only signal inputs to the ES register come from the exponent adder. When gES = 1, the adder outputs d_7 through d_0, are gated into ES_7 through ES_0 respectively. When -64gES = 0, the ES register is cleared to -64. As shown on the EAU drawing, the value to which each ES_i F-element is cleared is indicated by the side on which the -64gES signal enters.

The outputs, es_7 through es_0, feed the sEA selector and the EM register as described above. Outputs es_6 through es_0 are gated into $F0_{45}$ through $F0_{51}$ when RESgF0 = 1. When SEX (store exponent) is executed, the $\overline{es_7}$ signal is used to overwrite r_{39} through r_{44} causing es_7 to be gated into $F0_{39}$ through $F0_{44}$ when RESgF0 = 1. (See section 3.1.2.14.)

The es_7 and es_6 signals are used to generate $es_7 \oplus es_6$ as shown at the left center of the EAU drawing. This new signal is used only in the conditional logic of S9. It has unit value whenever the contents of ES lie outside the interval: $-64 \leq es \leq 63$. Any exponent outisde this interval is considered overflowed with respect to Flow Gate or Core Memory. Thus the "overflow" status memory element, OV, is set when $es_7 \oplus es_6 = 1$; the store order being executed is not logical, $\bar{\lambda} = 1$; and the number being stored is not zero (i.e., $\overline{RZ} = \bar{z} = 1$).

3.2.2.7 The sE Selector

The sE selector appears at the middle of the EAU drawing. The Boolean expressions for the outputs of this selector, E_i, are given below.

$$E_i' = em_i(EMsE) \qquad 0 \leq i \leq 7$$

The em_i signal is the true output of the EM_i F-element.

$$E_7 = E_7'$$

$$E_6 = E_6'$$

$$E_5 = E_5'$$

$$E_4 = E_4' \vee 22sE$$

$$E_3 = E_3'$$

$$E_2 = E_2' \vee 6sE \vee 22sE$$

$$E_1 = E_1' \vee 6sE \vee 22sE$$

$$E_0 = E_0'$$

As usual, when OsE is selected by DC, EMsE = 6sE = 22sE = 0. The 6sE setting is selected only in the P1 step of the P sequence when SIA is executed. The 22sE setting is selected by only two control steps: P1 for SIF, and D6 for any divide order.

3.2.2.8 The E Register

As shown at the center of the EAU drawing, the E register is composed of double-gated F-elements. The outputs of the sE selector, E_i, are gated into the E register when gE = 1. When DgE = 1, the outputs, d_i, of the exponent adder (D adder) are stored in the E register.

The d_i outputs can be gated into either the ES or E registers by turning on gES or DgE. If DgE = gES = 1, d_i is stored in both ES_i and E_i simultaneously (although somewhat more slowly). The arithmetic operations in the EAU are so arranged that DC never does this duplicate gating. Even if it should

become desirable at some future date, it is not recommended because the fan-out capability of the d_i would be overloaded in certain cases.

As indicated by the EAU drawing, the E register outputs, e_i, feed the sEA selector only.

3.2.3 The Exponent Adder (D-Adder)

The exponent adder (D-adder) shown on EAD (D-1503), operates over eight bits and has full carry assimilation. It is designed for high speed and large fanout. The contents of the EA register, ea, the output of the sD selector, eb, and the outputs of the "complement carry" status memory element, C, are its inputs. The sum of these inputs is represented by the d output in twos complement form. This output feeds the ES and E registers as well as the exponent decoder which is duscussed in the following section.

The adder is essentially composed of two ranks of modulo 2 adders and an 8-bit carry generator. The first rank of modulo 2 adders form $ea_i \oplus eb_i$. These signals together with others form the carry, c_i, into each position. The second rank of modulo 2 adders form the sum bits, $d_i = (ea_i \oplus eb_i) \oplus c_i$. For use in the exponent decoder the complement sum bits, $\overline{d}_i = (ea_i \oplus eb_i) \equiv c_i$, are also formed. Restoring logic is used to form both d_i and $\overline{d}_i$ to provide adequate fanout and voltage level at the input to the exponent decoder.

The Boolean expressions for the adder as shown on EAD are given below.

First Rank of Modulo 2 Adders

$$w_i = (ea_i)(\overline{eb}_i) \vee (\overline{ea}_i)(eb_i) \qquad 0 \leq i \leq 7$$

Carry Generation Logic

$$u_i = (ea_i)(eb_i) \qquad 0 \leq i \leq 4$$

$$v_i = (ea_i) \vee (eb_i) \qquad 0 \leq i \leq 6$$

In the expressions which follow, u_i and v_i may be further designated by -a or -b. This is the means used to distinguish between two signals which are logically identical but electronically different. For example, in the lower right hand corner of the EAD drawing u_0 - a and u_0 - b are equivalent logically but are formed with two separate nonrestoring AND circuits. This was done because the required fanout for the u_0 signal exceeded the maximum of three for a nonrestoring circuit. In the interest of speed, two nonrestoring AND's were used in place of one restoring AND.

$$u_1v_2 = u_1(v_2 - b)$$

$$u_{23} = u_2v_3 \vee u_3$$

$$v_2v_3 = (v_2 - a)v_3$$

$$w_3v_4 = w_3v_4$$

$$v_4v_5 = v_4v_5$$

$$u_{45} = u_4v_5 \vee (ea_5)(eb_5)$$

These miscellaneous signals are now used to form the carry into each position.

$c_0 = c$ (True output of the C status memory element)

$$c_1 = c_0v_0 \vee (u_0 - b)$$

$$c_2 = c_0v_0(v_1 - b) \vee (u_0 - b)(v_1 - b) \vee u_1$$

$$c_3 = c_0v_0(v_1 - b)(v_2 - b) \vee (u_0 - a)(v_1 - a)(v_2 - b) \vee u_1w_2 \vee u_2$$

$$c_4 = c_0w_0w_1(v_2v_3) \vee (u_0 - a)(v_1 - a)(v_2v_3) \vee (u_1v_2)v_3 \vee u_{23}$$

$$c_5 = c_1(v_1 - a)(v_2v_3)v_4 \vee (u_1v_2)(w_3v_4) \vee u_{23}w_4 \vee u_4$$

$$c_6 = c_2(v_2 - a)(w_3v_4)v_5 \vee u_{23}(v_4v_5) \vee u_{45}$$

$$c_7 = c_3(w_3v_4)w_5v_6 \vee u_3(v_4v_5)v_6 \vee u_{45}v_6 \vee u_6$$

Second Rank of Modulo 2 Adders with Complements

$$d_i = w_i \oplus c_i \qquad 0 \leq i \leq 7$$

$$\overline{d}_i = w_i \equiv c_i \qquad 0 \leq i \leq 7$$

To resolve the complex appearance of the carry generation logic, remove the "a" and "b" designations and substitute the expressions for the miscellaneous signals. The resulting expressions for the c_i contain minterms having a mixture of v_i and w_i variables. Replacing all of the v_i by w_i or vice versa would still yield a set of correct expressions for the c_i. This follows because a carry generated in the $i^{\underline{th}}$ position, i.e., $u_i = 1$, is propagated into the $(i-j)^{\underline{th}}$ position ($j > 0$) if either $w_k = 1$ or $v_k = 1$ for all k in the range: $i-j < k < i$. The possibility of using either w_i or v_i in the carry generation logic was pointed out by M. Faiman. The mixture of w_i and v_i signals that is used was determined largely by fanout limitations and the desire to keep the transistor cost at a minimum.

3.2.4 The Exponent Decoder

The logic for the exponent decoder is shown on the EDM drawing (D-1504). Its inputs are ea_7, eb_7, and the outputs of the exponent adder. In addition, the gEA, gES, and DgE signals are used to gate information into F-elements which indicate something about the range of the D adder output. The S7-d input is used to set the ES $\leq$ -64 F-element into the "1" state whenever the ES register is cleared to -64 during the S7 step of the store sequence.

With the exception of EDC, the F-elements shown on the EDM drawing form the "exponent decode" or ED register. The decoder outputs (e.g., d = -2) set the ED F-elements (e.g., ES = -2) when gED = 1. Note that gED = gES $\vee$ DgE.

The outputs of ES = -4, ES = 0, ES = 1, and ES = 2 are combined and

gated into EDC when gEA = 1. The EDC output, edc, is used in the conditional logic of the A9 control step. Both ES = -4, EDC, and the logic associated with their inputs were added to the original design of the decoder to provide slow and fast modes of operation for the add loop (steps A9 and A10). (See section 5.5.)

The outputs of the exponent decoder and the ED register are used primarily in the add loop (A9 and A10) and shift loop (F1 and F2). However, because the range of the D adder output is used so often to direct the action of DC, the decoder outputs are cabled to many other areas. Excluding the add and shift loops, one or more decoder or ED outputs are used in A1, A2, A3, A8, D7, D10, D13, F5, K2, M3, M5, M6, M7, P1, R2, S5, and S7.

The Boolean expressions for the decoder are given below. For the most part, these expressions were established by M. Faiman. The logic is developed as shown on the EDM drawing moving left to right from input to output. It is left for the reader to verify that the final outputs indicate the conditions their signal names imply, e.g., (d = 0).

First Row

$$(ea_7) \vee (eb_7) = \overline{(\overline{ea}_7)(\overline{eb}_7)}$$

$$(\overline{ea}_7)(\overline{eb}_7) = (\overline{ea}_7)(\overline{eb}_7)$$

$$\overline{(ea_7)} \vee \overline{(eb_7)} = \overline{(ea_7)(eb_7)}$$

$$(ea_7)(eb_7) = (ea_7)(eb_7)$$

} Cable Outputs

The numbers used to name the signals defined below (e.g., No. 21) have no significance with respect to values or range of values of d which the Boolean expressions define.

Second Row

No. 21 = d_7 (Used to increase fanout)

No. 42 = $\bar{d}_7$ (Used to increase fanout)

No. 16 = $\bar{d}_5 \vee \bar{d}_4\bar{d}_3 \vee \bar{d}_4\bar{d}_2$

No. 17 = $d_4 \vee d_3d_2$

No. 18 = $\bar{d}_4 \vee \bar{d}_3\bar{d}_2 \vee \bar{d}_3\bar{d}_1$

No. 22 = $\bar{d}_4 \vee \bar{d}_3\bar{d}_2\bar{d}_1 \vee \bar{d}_3\bar{d}_2\bar{d}_0$

No. 24 = $d_6 \vee d_5 \vee d_4d_3 \vee d_4d_2d_1$

No. 14 = $d_5 \vee d_4d_3 \vee d_4d_2 \vee d_4d_1d_0$

No. 25 = No. 28 = No. 29 = $d_7d_6d_5d_4d_3d_2$

No. 44 = d_7 (For fanout)

No. 15 = $\bar{d}_6 \vee \bar{d}_5\bar{d}_4\bar{d}_3\bar{d}_2\bar{d}_1\bar{d}_0$

No. 43 = $\bar{d}_7$ (For fanout)

No. 26 = No. 30 = $\bar{d}_7\bar{d}_6\bar{d}_5\bar{d}_4\bar{d}_3\bar{d}_2$

No. 31 = $\bar{d}_1$ (For fanout)

$\overline{(df = -1)} = \bar{d}_7 \vee \bar{d}_6 \vee \bar{d}_5 \vee \bar{d}_4 \vee \bar{d}_3 \vee \bar{d}_2 \vee \bar{d}_1 \vee \bar{d}_0$

The latter expression gives a "fast" indication of when d = -1. It is sensed only during the inner loop of the multiply sequence. Negative logic is used (i.e., $\overline{(df = -1)}$ = 0 when df = -1) to conform with the negative logic which is used throughout Delayed Control.

Third Row

No. 37 = (d = -1) = d_1d_0(No. 29)

No. 33 = $(d = -3) = \bar{d}_1 d_0$(No. 28)

No. 34 = $(d \leq -3) = \bar{d}_7 \vee d_1$(No. 28)

No. 39 = $(d = 0) = \bar{d}_0$(No. 26)(No. 31)

No. 36 = $(d = -2) = d_1 \bar{d}_0$(No. 29)

No. 9 = $(-45 d 44) = (-45 \leq d \leq 44) = \bar{d}_6$(No. 16)(No. 42)

$\vee \bar{d}_6 \bar{d}_4 \bar{d}_1 \bar{d}_0$(No. 42) $\vee d_6$(No. 14)(No. 44)

No. 35 = $(\overline{d \leq -2}) = \bar{d}_7 \vee d_1 d_0$(No. 29)

No. 32 = $(d = -4) = \bar{d}_1 \bar{d}_0$(No. 28)

No. 40 = $(d = 1) = d_0$(No. 30)(No. 31)

No. 41 = $(d = 2) = d_1 \bar{d}_0$(No. 30)

No. 38 = $(\overline{d \geq 2}) = d_7 \vee$ (No. 30)(No. 31)

$gED = \overline{(\overline{gES})(\overline{DgE})}$

Fourth Row

No. 31 = $(\overline{fa5}) = [(ea_7 \vee eb_7)]$(No. 21) $\vee \bar{d}_6$(No. 16)(No. 42)

$\vee [(ea_7)(eb_7)]$

fa5 = 1 implies that $d \geq 44$ This means that fa5 = 1 if d is overflowed, i.e., when the true value of d is ≥ 128.

No. 5 = $(\overline{fal}) = \bar{d}_7[\overline{ea}_7 \vee \overline{eb}_7] \vee d_7 d_6$(No. 14) $\vee [(\overline{ea}_7)(\overline{eb}_7)]$

fal = 1 implies that $d \leq -46$. This means that fal = 1 if d is underflowed; i.e., when the true value of d is ≤ -129.

No. 19 = $(\overline{fa4}) = d_6 \vee d_5$(No. 17) $\vee \bar{d}_5$(No. 18) $\vee$ (No. 21)

fa4 = 1 implies that $22 \leq d \leq 43$.

No. 23 = $(\overline{fa2,3}) = \bar{d}_6$(No. 21) $\vee$ $\bar{d}_5$(No. 21)(No. 22) $\vee$ $\bar{d}_7$(No. 24)

fa2,3 = 1 implies that $-45 \leq d \leq 21$.

These four signals are used in the A1 step of the add sequence to filter the appropriate case of floating point addition. (See section 5.5)

No. 11 = $\overline{dov}$ = [$(ea_7) \vee (eb_7)$] $\vee$ (No. 43)

No. 3 = dov = [$(\overline{ea}_7)(\overline{eb}_7)$](No. 44)

dov = 1 implies that $d \geq 128$.

No. 10 = dz = [$(ea_7)(eb_7)$](No. 43)

No. 7 = $\overline{dz}$ = [$(\overline{ea}_7) \vee (\overline{eb}_7)$] $\vee$ (No. 44)

dz = 1 implies that $d \leq -129$.

No. 12 = $(d \leq -64)$ = [$(ea_7) \vee (eb_7)$](No. 15)(No. 44) $\vee$ [)ea_7)(eb_7)](No. 43)

$(\overline{d \geq 0})$ = d_7(No. 11) $\vee$ (No. 10)

When gED = 1, many of the decoder outputs defined above are gated into the corresponding F-elements of the ED register. For example, No. 8 = dov is gated into the ESOV F-element when gED = gES $\vee$ DgE = 1. If the true output ("1" side) of these F-elements has unit value (positive voltage) following this gating operation, then the condition indicated by the F-element name is true with respect to the contents of ES or E depending on whether gES or DgE was performed. The conditions indicated by the ED register refer to the ES register only because gES is used a little more frequently than DgE. When gED = 1 because DgE = 1, these conditions should all read E = -1, E = -3, etc., instead of ES = -1, ES = -3, etc.

Fifth Row

This row consists primarily of F-elements and cable drivers. One exception is the logic used to form the input to the EDC F-element.

$$EDC = (\overline{es = -4})(\overline{es = 0})(\overline{es = 1})(\overline{es = 2})$$

This signal is gated into EDC when gEA = 1. As explained earlier, the true output of EDC, edc, is used only to control the transition between slow and fast modes of shifting in the A9 step of the floating add sequence.

The dov and dz signals deserve special attention. Since the exponent adder operates modulo 256 with the most significant bit position assigned a weight of -2^7, its output, d, may be overflowed in the positive direction (dov = 1) even though d appears negative (i.e., $d_7 = 1$) and conversely. Let d = ea + eb + c where c = 1 or 0 depending on whether C is true or not. In the following case analysis, C will be assumed true or false as required to establish the limits.

(A) If $0 \leq ea \leq 127$ and $0 \leq eb \leq 63$, then

$0 \leq d \leq 127$ or $-128 \leq d \leq -65$.

(B) If $0 \leq ea \leq 127$ and $0 \leq eb \leq 127$, then

$0 \leq d \leq 127$ or $-128 \leq d \leq -1$.

Note that in both cases the sum overflowed when d fell in a negative range. To detect these cases, the decoder must note that ea and eb are both positive and d is negative. Hence, dov = $(\overline{ea_7})(\overline{eb_7})(d_7)$ which agrees with the equation given above.

(C) If $-128 \leq ea \leq 0$ and $-64 \leq eb \leq 0$, then

$-128 \leq d \leq 0$ or $64 \leq d \leq 127$.

(D) If $-128 \leq ea \leq 0$ and $-128 \leq eb \leq 0$, then

$-128 \leq d \leq 0$ or $0 \leq d \leq 127$.

In these two cases the sum is underflowed (i.e., overflowed in the negative direction) when d falls in a positive range. When d is underflowed, the sum

represented by the fractional and exponential parts is considered zero. It is clear that dz must have unit value when ea and eb are both negative and d is positive. Thus, $dz = (ea_7)(eb_7)(\bar{d}_7)$ which agrees with the equation given above. Note that neither dov nor dz can be true if ea and eb disagree in sign regardless of how C is set.

Similar considerations must be taken into account when analyzing the formation of the $(d \leq -64)$ and $(\overline{d \geq 0})$ signals. These considerations were overlooked in the original design. This was pointed out by R. H. Farrell and subsequently corrected as shown above.

One additional comment concerning the timing of the gES and DgE gates is in order at this point. When the exponent adder is being used as a counter to control the number of steps performed in the add, shift, multiply, or divide loops, the flow of information through the adder and decoder may be described as follows. If the inputs to the exponent adder are changed either by gating a new word into EA or by changing the setting of the sD selector, some time must elapse before the adder output is reliable. Additional time is needed before the decoder outputs are reliable. Therefore, the adder output, d, could be gated into ES or E via gES or DgE before the decoder outputs are reliable at the inputs to the ED register. However, since $gED = gES \vee DgE$, it is necessary to insure that gES or DgE remains on long enough for the decoder output to become reliable and set the ED F-elements accordingly.

4. THE LINK MECHANISMS

The Link Mechanisms (LM) include gates, selector mechanisms, and control status memory elements. Delayed Control (DC) directs the data flow and processing in the Arithmetic Unit (AU) via the LM. DC requests may determine the state of the LM directly or indirectly through the outputs of decoders. The inputs to these decoders may be the outputs of registers, adders, or status memory elements. Selector mechanism and status memory elements remember their present state. The outputs of certain status memory elements influence the branching of DC. The setting of one or more of these elements during the present control step partially determines the sequence of future control steps.

If the design of DC were completely speed-independent, each request would be answered with a reply of the same parity indicating the action requested has been completed. The goal of speed-independence was only partially realized. With few exceptions, a direct or indirect request to a link mechanism is answered with a reply of the same parity which indicates that at least part of the link mechanism logic has responded. This reply does not indicate completion of the resulting change in state of the AU and/or control path. DC also sends bypass requests to certain LM to obtain a reply without changing the LM output.

In terms of physical location, the LM occupy a central position in the Arithmetic Subsystem. They collect requests from DC and send their outputs to the AU and their replies back to DC. The gates and selector mechanisms are located in the Q, A, and S elevels of bays 6C, 7C, 8C, 16FC, and 16RC. The control status memory elements are primarily located in Q8F and Q8R. Certain specialized status memory elements such as EXA, EXF, and EXR are located with the associated control area logic.

To facilitate the collection of requests and the distribution of replies, four request areas (generally covering one A level chassis) and thirteen reply areas were established during layout of DC. The requests from various

control areas are fed to secondary AND's (actually $\overline{A}'$s followed by inverting cable drivers) located in one of the four request areas. The outputs of these secondary AND's then feed a primary AND next to the prime driver(s) or the inputs of the link mechanism being considered. The reply signal is distributed by one to three cable drivers to NOT circuits located in the various reply areas. The reply signals at the output of these NOT's then feed the $\overline{A}^*$'s and OR's in the appropriate control areas.

The four request areas are: RQ-1 located in the A9F chassis, RQ-2 located in A15F, RQ-3 located in A15R and part of A14R, and RQ-4 located in A10R. The thirteen reply areas are: RP1-1 in Q9F; RP1-2 in S9F; RP2-1 in Q17 and Q16F; RP2-2 in S16F and S15F; RP2-3 in A15F and Q16F; RP2-4 in A17, A16R, S17, and S16R; RP3-1 in Q16R and Q15R; RP3-2 in S15R and S14R; RP3-3 in A14R and Q14R; RP4-1 in Q9R and Q10R; RP4-2 in S10R and S11R; RP4-3 in A11R and Q11R; and RP4-4 in A9R and S9R.

Request and reply references are shown in the input and output tables at the left and bottom of the LM drawings. Unfortunately, these tables differ from those appearing on AU and DC drawings. A key for the information contained in these tables appears in the upper and lower left-hand corners of all LM drawings. For additional information regarding the layout of the LM and the associated logic, see File No. 528, "A Description of the Logic Drawings for the Arithmetic Subsystem" by J. O. Penhollow.

A word of caution is in order concerning the signal notation for requests, replies, and outputs of LM. Negative logic notation is used in DC and positive logic notation is used in the AU. For example, D10-a or gA-4 (D-1187) will turn gA "on" when they have a negative voltage level, i.e., D10-a = 0 or gA-4 = 0. The gA outputs shown at the top of the drawing are "on" when they have a positive voltage, i.e., when gA = 1. If gA = 1, the outputs of the sA selector as described in section 3.1.2.6 are allowed to set the A register F-elements. In response to a "0" request by D10-a or gA-4, a "0" reply such as gAr-c or gAr-cla is returned to DC. When all of the LM requested by a given

control area have returned "0" replies, the control area responds by sending out a "1" request signal which turns off gates (so that gA = 0) and places the selector mechanisms and status memory elements in a memory state, i.e., allows them to retain the state to which they were set by the "0" request. The LM replies respond to a "1" request by returning a "1" to DC which conditions the next control area. Although a gate was used as a specific example, similar statements apply to the requests, replies, and outputs of selector mechanisms and status memory elements.

In the next three sections the logical operations of a sample gate, selector mechanism and control status memory element are discussed. Special LM are also considered.

4.1 Gates

There are seven regular gate logic drawings: gA (D-1187), gQ (D-1188), gS (D-1189), gR (D-1190), gEA (D-1193), gE (D-1195), and ESgEM (D-1500). The operation of gQ is discussed as an example.

Assume that gQ is off initially such that all request inputs at the left side of the drawing are "1", all replies (i.e., NOT outputs) at the bottom of the drawing are "1", and all gQ outputs at the top of the drawing are "0". Assume that an MPY order is being executed and control area M3 (D-1281) has just become active. The M3-a output goes from "1" to "0". This request is fed to the secondary AND cable driver in RQ-2 (A16F). Its output, gQ-2, goes from "1" to "0" and feeds the primary AND in the A8C chassis. The output of this AND, RQ-gQ also goes from "1" to "0". This causes gQ to go from "0" to "1" at the gate inputs of the F elements in the Q register. Under slow reply conditions, the change from "1" to "0" at the otuput of the $\overline{D}^2$ in Q5C is fed to the INHIBIT-AND-OR in A8C causing the output of this element to go from "1" to "0" provided the MSAQ inhibit signal from the console is "1". The "0" output of the INHIBIT-AND-OR feeds three inverting cable drivers and a collector follower. The cable drivers feed seven NOT circuits in various reply areas. In particular, the gQR-bla reply goes from "1" to

"0" and feeds the $\overline{A_1^*}$ and OR circuit associated with control area M4 (D-1281).

When all reply inputs to the two OR's in M4 are "0", the output of the second OR causes the Eccles-Jordan in M3 to set to a "0" so that the associated M3-a output changes from "0" to "1". By following this "1" through the gQ logic, it is easily seen that gQ goes from "1" to "0", thus closing the gate to the Q register while the replies including gQr-bla go from "0" to "1". When all of the reply inputs to the $\overline{A_1^*}$ in M4 are "1" its output goes to "0" since the M4 Eccles-Jordan was set to "1" while M3-a was "0". This represents the beginning of the next control step in the multiply sequence.

Under fast reply conditions, the gQr input to the INHIBIT-AND-OR is disconnected and the output of the primary AND, RQ-gQ, is connected in its place. This eliminates five collector delays in the feedback loop but sacrifices a check on the prime gate driver and one of the $\overline{D^2}$ drivers. Note that the exponent gate logic always operates under fast reply conditions.

The MSAQ inhibit input is controlled from the console and is used during step-by-step sequencing of DC. As long as MSAQ = 1, the gQ replies are allowed to go from "1" to "0" in the normal manner. If MSAQ = 0, the gQ replies are held on a "1" until the operator removes the inhibit causing MSAQ = 1. When a control area requests gQ and MSAQ = 0, the Q register F-element gates are held open. Likewise, any other LM requested by this control area are held in the active state. The Eccles-Jordan associated with the control area cannot be set to "0" and thus allow the $\overline{A_1^*}$ to produce a "1" request until the gQ reply is allowed to change from "1" to "0". For additional information concerning inhibit signals, see File No. 409, "Manually Stepping Through DC" by R. E. Swartwout.

In certain control steps gQ is done conditionally. The logical structure of DC is such that a reply from the gQ logic is needed in these cases whether the gate is performed or not. A bypass gQ request is used to obtain the reply when the gate is not performed. Note that if either By gQ-2 or By gQ-4 go from "1" to "0", the gQ replies (such as gQr-ala, etc.) go from "1" to "0"

but gQ at the inputs to the Q register F-elements remains "0".

In the remainder of this section, we consider four gate logics which have some irregularites and six special gates which do not appear on separate drawings.

The DgE and gES drawings show an output to EDM (D-1504). When either DgE or gES is requested, the gED gate shown on EDM is also requested. Note that gED = gES $\vee$ DgE as discussed in section 2.2.4.

The -64gES gate logic also appears on gES (D-1194). Since this signal feeds the clear inputs of the ES register F-elements, it is active when -64gES is "0". Therefore, a LEVEL RESTORER appears in place of a NOT on Q7R. A "0" request for -64gES produces a "0" gES reply in DC as indicated by the AND ahead of the cable drivers in A8C.

The FlgMEM and RESgOUT logic appears on FRgMF (D-1194). The former is used only during decode (G1) while the latter is used only during store (S9). The collection of requests and the distribution of replies is limited accordingly. The primary gate drivers are located in the G1 (D-1276) and S9 (D-1284) control areas.

The R $\cdot$ MgM and R $\vee$ MgM gate logic is shown on RgM (D-1191). If both R $\cdot$ MgM and R $\vee$ MgM are requested similtaneously, RgM is accomplished as discussed in section 3.1.2.2. During the clear add, (B), sequence, R $\cdot$ MgM or R $\vee$ MgM may be done separately. The conditional logic that decides which gate is actually performed during B2 appears at the left edge of the RgM drawing. B2-RgM must be "0" before any output of this logic can assume a value other than "1".

The a_0gA_{-1} gate is shown on HAS (D-1522). It is used to copy the contents of A_0 into A_{-1} during the execution of a shift instruction, a store instruction which uses the shift, (F), sequence, or a BLS order. The black gate symbol by the A_{-1} F-element indicates that a_0gA_{-1} is "0" when active.

The $0gQ_{43}Q_{44}$ gate logic appears on LQR (D-1524). This gate clears the Q_{43} and Q_{44} F-elements to zero during decode (G1) and K3 step at correct overflow,

(K), sequence when clearing an SSC or ASC order. The $0gQ_{43}Q_{44}$-4 signal is "0" when active.

The $m_{44}gEM_7$ gate is used only during E1 of the exponent arithmetic, (E), sequence. The gate driver appears in the E1 control area (D-1270). This gate allows eight-bit exponents to be brought into the EAU from F1. When $m_{44}gEM_7 = 1$, the contents of M_{44} are copied into EM_7. This overwrites the $F1_{45}$ bit that was placed in EM_6 and EM_7 during G1 via F1gMEM. (See section 3.2.2.1.)

The gCB logic appears in the carry-borrow section of SEC (D-1527). When gCB = 1, the output of the carry-borrow logic is transferred to the CB F-element (see section 3.1.13). As shown on SEC, this gate is only used during the A11 step of the add, (A), sequence and the D13 step of the divide, (D), sequence.

The gH_1' gate request occurs only during the D6 control step of the divide, (D), sequence as shown on D-1272. The gate driver and the H_1' Eccles-Jordan appear on DPα (D-1507). When $gH_1' = 0$, H_1' is set to "1". This is done to insure proper recoding of the first quotient digit as discussed in section 3.1.17.

4.2 Selector Mechanisms

There are seven drawings of selector mechanism logic. Four selector mechanisms are associated with the MAU: MsA-I (D-1176), MsS-I (D-1178), sAQ-I (D-1180), and sSR-I (D-1182). Their companion drawings showing the distribution of selector gate signals to the selectors in the MAU are: MsA-II (D-1177), MsS-II (D-1179), sAQ-II (D-1181, and sSR-II (D-1183). The three selector mechanism drawings associated with the EAU are: sD (D-1184), sEA (D-1185), and sE (D-1186). These drawings also show the distribution of the selector gate signals.

In all cases, the selector may be viewed as an n-state Eccles-Jordan

memory element where n is the number of selector options. Certain peculiarities arise in connection with the MsS selector mechanism because the $M\bar{M}sS$ request causes the MsS and $\bar{M}sS$ options to be turned on together. This is discussed in detail below. The basic logic for the selector mechanisms was suggested by Professor D. E. Muller. It was developed and modified to fit the needs of the order code and a "speed-independent" DC by R. E. Swartwout.

Each active request ("0" input) causes the selector mechanism to release its old setting and check that all selector gate signals (outputs) are "0", and then establish and latch the new setting. These events occur sequentially to prevent overloading the selector emitter followers at the associated register or adder output. If the old and new selector settings were on simultaneously, the emitter current into these transistors would be twice the design value. After the selector mechanism has theoretically latched in its new state, the associated reply goes from "1" to "0". When this indication is received by the control area following the one which initiated the selector request (i.e., the active control area), it is mixed with other replies in an OR. When the output of this OR goes from "1" to "0" to the Eccles-Jordan in the active control is set to "0", causing the request at the $\bar{A}^*$ output to change from "0" to "1". This returns the active input to the selector mechanism to "1" and leaves the state of the selector unchanged, but causes the reply to go from "0" to "1". When all activated link mechanisms are relaxed such that their replies are "1", the output of the $\bar{A}^*$ in the following control area goes from "1" to "0" which initiates the next control step.

The method of collecting requests and distributing replies is similar to the one described for the gates. It should be noted that many of the selector requests are made by decoders which are conditioned by certain control area $\bar{A}^*$'s. Examples of these are the μ and θ decoders and the division predictors shown on $\mu\theta D$ (D-1506) and DPα (D-1507). The use of bypass requests and console inhibit

signals is the same as described for the gates. Connections for fast and slow replies also exist for the selector mechanisms associated with the MAU. As for the gates, the shorter reply time is achieved by eliminating the prime driver, a $\overline{D}^2$ driver, and a return cable driver from the reply chain. The fast reply connections are indicated by dotted lines on sAQ-I (D-1180) and sSR-I (D-1182). The replies for the EAU selector mechanisms are all fast in this sense.

As an example of how the selector mechanisms operate, consider MsA (D-1176). Assume MsA is set to OMsA initially, meaning that the b_i outputs of the MsA selector in the MAU are all "0". Let MSAQ = 1 (i.e., the console inhibit switch is "off"). Furthermore, assume the inputs to the restoring AND's shown down the center of the drawing are all "1", and the replies at the outputs of the NOT's across the bottom of the drawing are also all "1". It follows that RQ-2MsA = RQ-MsA = RQ-OMsA = RQ-$\overline{\text{M}}$sA = RQ-KgA = RQ-MsA-a = RQ-MsA-b = 1. At the top of the drawing, 2MsA = MsA = $\overline{\text{M}}$sA = KgA = $\overline{\text{2MsAr}}$ = $\overline{\text{MsAr}}$ = $\overline{\overline{\text{M}}\text{sAr}}$ = $\overline{\text{KgAr}}$ = $\overline{\text{OMsA}}$ = OMsAr = 0, while OMsA = $\overline{\text{KI}}$ = 1. At the right side of the drawing, we have RP-2MsA = RP-MsA = RP-OMsA = RP-$\overline{\text{M}}$sA = RP-KgA = 1. If D14-ByMsA = ByMsA-4 = 1 (i.e., these bypass requests are not active), then $\overline{\text{MsAr-a}}$ = $\overline{\text{MsAr-b}}$ = 0. The direct reply to M6 and D11, MsAr-b, and the NOT outputs are all "1". At the bottom of the selector mechanism logic, KgA = ca = 0 and $\overline{\text{KgA}}$ = 1. The destinations of these outputs as well as the reply, MsAr-b, are shown in the "Direct Connection" boxes at the bottom of the drawing.

Under the above conditions, the selector gate signals denoted by 2MsA, MsA, $\overline{\text{M}}$sA, and KgA are held to "0" by the OMsAr which is also "0". Note that OMsAr feeds each of the four nonrestoring AND's which appear in a line down the center of the MsA-I drawing.

Now suppose that one of the following $\overline{\text{M}}$sA request inputs goes from "1" to "0": $\overline{\text{M}}$sA-μ1, $\overline{\text{M}}$sA-μ2, $\overline{\text{M}}$sA-θ, $\overline{\text{M}}$sA-ρ, or D14-$\overline{\text{M}}$sA. The source of these requests

is indicated by the input table. RQ-$\overline{\text{M}}$sA goes to "0" which in turn causes RQ-MsA-b to go to "0". RP-$\overline{\text{M}}$sA remains a "1" momentarily since $\overline{\text{M}}$sAr is still "1" at this point. 2MsA, MsA, $\overline{\text{M}}$sA, and KgA remain at "0" and $\overline{\text{KI}}$ remains at "1" because OMsAr is still "0". RQ-MsA-b going to "0" changes OMsA from "1" to "0" and causes $\overline{\text{OMsA}}$ = OMsAr to change from "0" to "1". The old setting has been released at this point. Note that RP-OMsA and all other such signals are still "1". When OMsAr goes to "1", 2MsA, MsA, and KgA remain at "0" and $\overline{\text{KI}}$ remains at "1" since RQ-$\overline{\text{M}}$sA = RQ-MsA-b = 0. However, $\overline{\text{M}}$sA goes from "0" to "1". As shown on MsA-II (D-1177), $\overline{\overline{\text{M}}\text{sAr}}$ also goes from "0" to "1" causing the $\overline{\text{M}}$sAr NOT output on MsA-I (D-1176) to go from "1" to "0". Since RP-$\overline{\text{M}}$sA is still "0", RP-$\overline{\text{M}}$sA also goes from "1" to "0" and causes all of the replies to change likewise. At the same time, the "0" $\overline{\text{M}}$sAr signal is fed back to the nonrestoring AND's whose outputs are RQ-MsA-b and KgA. This is the latch signal which holds the $\overline{\text{M}}$sA selector setting when RQ-$\overline{\text{M}}$sA goes back to "1". The latter change occurs when the "0" reply in conjunction with "0" replies from other active link mechanisms causes the request from the active control area to change from "0" to "1". When this causes RQ-$\overline{\text{M}}$sA to go from "0" to "1", RP-$\overline{\text{M}}$sA changes from "0" to "1" and causes all the reply outputs to follow. This completes the active cycle of the selector mechanism.

If the next request to the MsA mechanism is also for $\overline{\text{M}}$sA, the MsA reply changes from "1" to "0" much faster. Since $\overline{\text{M}}$sAr = 0, RP-$\overline{\text{M}}$sA changes to "0" as soon as RQ-$\overline{\text{M}}$sA does. This is simply a consequence of the fact that the selector mechanism is already in the state requested. If the next request is not $\overline{\text{M}}$sA, a process similar to the one described above will yield the new setting.

It is worthwhile to note that $\overline{\text{M}}$sA and ca have the same parity. The ca signal is the carry input in the $44^{\underline{\text{th}}}$ position of the A adder as shown on LAS (D-1523) and LAD (D-1212). In complement arithmetic, the subtraction of M from A

is accomplished by adding $\overline{M}$ (bitwise complement) to A with a carry in the least significant position. Therefore, M is subtracted from A when $\overline{M}sA = ca = 1$. (See section 3.1.5.)

The delay shown in the RP-KgA output line is necessary to insure that the assimilated value of A appears at the A-adder output before starting the next control step. This is especially important if the next step involves a control branch which is conditioned on the α-logic outputs shown on DPα (D-1507), and discussed in section 3.1.9. An example of this is the K1 control step shown on D-1279. Since KgA is usually requested by the same control area that requests gA, the time required for the A-adder outputs to become reliable is quite long. The A register must set to its new value feed the carry generator which in turn must feed the A adder. There is no way of knowing when the A adder outputs are reliable so the reply must be sufficiently slow to cover the worst case condition.

Because of its location, the delay in the RP-KgA output line only needs to be half of the total delay required under worst case conditions. When KgA is requested, the "1"-"0" and "0"-"1" changes on RP-KgA are both delayed by the amount specified even though KgA may have been set previously. A delay is necessary under the latter condition because all of the stored carries gated into A at the time of the second KgA request are "0". This change in A must flush through the carry generator and A adder before the output of the latter is reliable. When the second request is for any other setting of the MsA mechanism, the reply time is not influenced by the KgA delay. Furthermore, this delay does not slow down the KgA latch-back signal, KgAr.

The KgA and $\overline{KI}$ signals shown at the top of D-1176 are logical complements. When KgA = 1, the non-stored carry outputs of the A adder represent the assimilated value of the A register. The stored carry outputs, α_1^*, would be

equal to the even-numbered b_i if allowed to operate normally. Since these carries have already been taken into account by the adder, the $\overline{KI}$ = 0 signal is used to inhibit them. Therefore, $\alpha_i^* = 0$ for all i when KgA = 1.

The MsS selector mechanism logic (D-1178) is complicated by the need to set MsS and $\bar{M}$sS simultaneously with cs = 0 when M$\bar{M}$sS is requested. This causes a field of units to be added to the contents of the S register which amounts to subtracting a unit in the $44^{\underline{th}}$ position. The NOT order which does a bitwise complement of the operand in M requires that $\bar{M}$sS be set with cs = 0. Furthermore, the SUB, SSC, CSB, CST, STN, MPY and all divide orders use either $\bar{M}$sS or $\overline{2M}$sS with cs = 1. During the terminal steps of a case 4 addition and during quotient round-off, it is convenient to allow cs = 1 with OMsS = 1. This corresponds to the CS request which adds a unit in the $44^{\underline{th}}$ position.

The speed-independent realization of the MsS selector mechanism uses two Eccles-Jordans called MO and CS. There is no restriction on the order of the requests except a CS or M$\bar{M}$sS request must always be followed by the clear request, KMsS. The latter returns the mechanism to the OMsS state and insures that MsS and $\bar{M}$sS selector gate signals are both "0".

There are four selector replies: MsSr, M$\bar{M}$r, CSr, and KMsSr. The MsSr signal corresponds to MsAr and is used most frequently. The remaining replies are used in pairs. When 2MsS; MsS; OMsS; $\bar{M}$; $\overline{CS}$; $\bar{M}$sS; or $\overline{2M}$sS is requested, MsSr changes from "1" to "0" while M$\bar{M}$r, CSr, and KMsSr all remain at "1". If CS is requested, only CSr changes from "1" to "0". If M$\bar{M}$sS is requested, only M$\bar{M}$r and CSr go from "1" to "0". When KMsS is requested, only KMsSr and CSr go from "1" to "0". The By-MsS, By-M$\bar{M}$, and By-CS requests also cause the corresponding replies to change from "1" to "0". There is no console inhibit associated with M$\bar{M}$r, CSr, or KMsSr.

The MO memory element has a reply output and is normally in the "1"

state. It is set to "0" only when $M\bar{M}sS$ is requested. The CS memory element is set to "0" except after a $\bar{M}sS$, $\bar{2}\bar{M}sS$, or CS request.

As an example of how the MO and CS memory elements function, consider an $M\bar{M}sS$ request followed by KMsS. As in the MsA example, assume the mechanism is initially set to OMsS and all requests and replies are "1". MO is set to "1" (i.e., mo = 1) and CS is set to "0".

An active $M\bar{M}sS$ request causes RQ-$M\bar{M}sS$ to change from "1" to "0". This in turn changes RQ-$KM\bar{M}sS$ from "1" to "0" which yields CSr = 0 since cs = 0 initially. At the same time, MO is set to "0" such that mo = 0 and MOr = 0. Since $\overline{mo}$ goes to "1" before MOr goes to "0", KMsSr remains at "1". The output of the nonrestoring OR which has RQ-$M\bar{M}sS$ and mo as inputs (appearing a little left of center on D-1178) now changes from "1" to "0". The output of the nonrestoring AND which it feeds changes in a like manner and causes OMsS to go from "1" to "0" while OMsSr goes from "0" to "1". The latter output feeds the nonrestoring AND directly to its left and causes its output to change from "0" to "1". This AND output then causes both MsS and $\bar{M}sS$ to change from "0" to "1" since $\overline{mo}$ = 1. The resulting replies -- MsSr and $\bar{M}sSr$ -- go from "1" to "0". This does not change RP-2MsS; RP-$\bar{2}\bar{M}sS$; RP-OMsS; RP-MsS; RP-$\bar{M}sS$; RP-$\bar{M}sS$, $\overline{CS}$; or KMsSr which have remained at "1" during the preceding action. However, MsSr = $\bar{M}sSr$ = MOr = mo = 0 causes $M\bar{M}r$ to go from "1" to "0". The "0" MsSr and $\bar{M}sSr$ signals also feed the nonrestoring AND having (mo) $\vee$ (RQ-$M\bar{M}sS$) as an input. This latches the $M\bar{M}sS$ state. When RQ-$M\bar{M}sS$ goes from "0" to "1", CSr changes from "0" to "1". Furthermore, MOr returns to "1" causing $M\bar{M}r$ to follow.

KMsS must be the next request. Incidentally, this request is used when DC is cleared. When RQ-KMsS goes to "0", RQ-$KM\bar{M}sS$ also goes to "0" and sets CS to "0". (The latter operation is necessary in case CS was the previous request.) This yields CSr = 0. At the same time, RQ-KMsS causes MsS and $\bar{M}sS$ to change to "0", and holds 2MsS and $\bar{2}\bar{M}sS$ on "0" while MsSr and $\bar{M}sSr$ change from

"0" to "1". The latter changes permit OMsS to go to "1" with OMsSr changing to "0", thus latching the new state. Since OMsSr = RQ-KMsS = 0, MO is now set to "1" such that $\overline{mo}$ = MOr = 0. This causes KMsSr to change from "1" to "0". When RQ-KMsS returns to "1", RQ-K$\overline{M}$sS and CSr go to "1". Furthermore, RQ-MO goes to "1" which causes MOr and KMsSr to change from "0" to "1". The mechanism is left in the OMsS state -- meaning that the t_i outputs of the MsS selector in the MAU are all zero.

In the sAQ or sSR mechanisms a state may correspond to different settings for the associated A and Q or S and R selectors. In all cases, the gate signals are defined for sA and sQ or sS and sR selectors in the MAU. For example, the sSR mechanism shown on D-1182 has a state called OAsSR. In this state, the sS and sR selectors in the MAU are set to OsS and AsR respectively. When gS = gR = 1, zeros are placed in the S register and the output of the A-adder, α, is placed in the R register.

The sD mechanism (D-1184) is similar to the MsA and MsS mechanisms except that an $\overline{EM}$sD request does not cause a complement carry to be added to the least significant position of the exponent adder. This request sets the sD mechanism to the $\overline{EM}$sD state which insures that $eb_i = \overline{em}_i$ as shown on EAU (D-1502). If the contents of EM are to be subtracted from the contents of EA, $\overline{EM}$sD and C are set to "1". The true output of C (D-1285) represents the complement carry into the $0^{\underline{th}}$ position of the exponent adder as discussed in section 3.2.3.

Many of the states in sD, sE, and sEA represent constants at the outputs of their associated selectors. For example, 2sD produces $eb_i = 0$ for $i = 0$ and $2 \leq i \leq 7$ while $eb_i = 1$. Since the $1^{\underline{st}}$ bit position in the EAU has a weight of 2, the effect of this selector setting is to add 2 to the contents of EA. As a second example, 6sE causes the $1^{\underline{st}}$ and $2^{\underline{nd}}$ inputs to the E register to be "1" while all others are "0". Thus, when the sE mechanism is in this state and gE is perform, the E register is set to 6.

4.3 Control Status Memory Elements

The control status memory elements are Eccles-Jordans with reply outputs. The circuit schematic and logical structure of this element is shown on drawings B-954 and D-1136. If EJ is a control status memory element, the request inputs to the "1" and "0" sides are called EJ and $\overline{EJ}$ respectively. The reply is called EJr. The outputs from the "1" and "0" sides are called ej and $\overline{ej}$ respectively. In the memory state, EJ = $\overline{EJ}$ = 1 such that ej and $\overline{ej}$ retain their previous values while EJr = 1.

Assume that ej = 1 and $\overline{ej}$ = 0. If $\overline{EJ}$ now goes to "0" while EJ = 1, $\overline{ej}$ changes to "1" while ej and EJr remain at "1". After $\overline{EJ}$ = 0 and $\overline{ej}$ = 1, ej goes to "0" causing EJr to change from "1" to "0" since EJr = $(EJ \vee \overline{ej})(\overline{EJ} \vee ej)$. The "0" reply indicates the memory element has been set to the new state and theoretically latched. When $\overline{EJ}$ goes back to "1" with the EJ input remaining at "1", the new state is retained and EJr goes to "1". This completes an active cycle for the Eccles-Jordan with reply. As in the case of selector mechanisms, the EJ reply goes to "0" much faster when the Eccles-Jordan is already in the state requested.

The method of collecting requests and distributing replies and outputs is similar to that described for gate and selector mechanisms. As opposed to the other types of link mechanisms, some control status memory element outputs feed logic in both the AU and DC. By feeding the reply output to an AND, bypass requests are possible. These requests enter the AND, causing its output to change from "1" to "0" to "1" without changing the state of the memory element.

All except one of the control status memory elements (these do not include DCRFL, INFL, and DCRS) are set to both "0" and "1" by DC. The exception is OV (D-1291) which is set to "1" by DC but is reset to "0" (i.e., cleared)

by AC. OV is also the only status memory element whose reply may be inhibited from the console. This is accomplished by making MSOV = 0.

A brief description of the function of each control status memory element follows.

The C status memory element appears on D-1285. It is conditionally set to "0" or "1" during G1 (D-1277). During the execution of DC orders, it is set to "1" whenever a carry into the least significant position of the exponent adder is needed. The c output represents this carry as shown on EAU (D-1502) and discussed in section 3.2.3. The c and $\overline{c}$ signals are also used as inputs to the conditional logic of the K2 (D-1279) and R2 (D-1282) control areas.

CL and CR shown on D-1286 are used during the circular left and right shift operations for case 3 addition as discussed in sections 3.1.12 and 5.5. CL is set to "1" during A9 (D-1269) to condition the paths $\alpha_{-1} \rightarrow R_{43}$, $\alpha_0 \rightarrow R_{44}$ and $\sigma_{-1} \rightarrow Q_{43}$, $\sigma_0 \rightarrow Q_{44}$. It is set to "0" during A11. Likewise, CR is set to "1" during A10 (D-1269), to condition the paths $\alpha'_{-3} \rightarrow S_{-1}$, $\alpha'_{-2} \rightarrow S_0$ and $\sigma'_{-3} \rightarrow A_{-1}$, $\sigma'_{-2} \rightarrow A_0$. CR is set to "0" during G1 (D-1276), K1, or K3 (D-1279). The cr and $\overline{cr}$ outputs are also used as inputs to the conditional logic of A9 and A10. To increase the speed of the add loop (A9 and A10) bypass logic for the CL and CR replies is also provided.

The DL memory element shown on D-1287 is set to "1" to condition the paths $\nu_{-1} \rightarrow S_{43}$, $\nu_0 \rightarrow S_{44}$ and $\rho_{-1} \rightarrow A_{43}$, $\rho_0 \rightarrow A_{44}$. DL is conditionally set to "0" or "1" during G1 (D-1277) as well as D3 and D10. It is always set to "1" during A13, D6, and F3, and is always set to "0" during A6 and D16. The dl and $\overline{dl}$ outputs also feed the conditional logic in the A9, A11, D5, D11, F1, F2, F8, K1, K2, R1 and R2 control areas. In many cases, dl and $\overline{dl}$ determine whether gQ and gR are performed or bypassed. If dl = 1 (i.e., if DL is true), a double length left shift in S,R and A,Q is possible, so gS,gR and gA,gQ are used. If

d1 = 0, only a single length left shift is S and A is possible, so gS, By gR and gA, By gQ are used.

The EXA, EXF, and EXR status memory elements shown on D-1288 control the exit from the inner loops of the (A), (F), and (R) sequences respectively. The exa and $\overline{exa}$ signals feed the $\overline{A}^*$s of A11 and A9 (D-1269) respectively. The $\overline{A}^*$s of F3 and F1 (D-1274) have exf and $\overline{exf}$ as inputs. Likewise, exr and $\overline{exr}$ feed the $\overline{A}^*$s in R3 and R1 (D-1282) respectively. Note that EXA and EXR are set to "0" when DC is cleared. EXF is set to "0" by the G1 request which sets the F1 Eccles-Jordan to "0".

In general, the EX memory element is set to "0" or "1" prior to entering the associated control loop. If ex = 1 at the time of entry, the loop is bypassed. For example, this may occur when entering the normalize, (R), loop. If exr = 1, upon entry from G1, control passes immediately to R3 and on to the first control step in the (D), (M), (S), or (V) sequences. If ex = 0 at the time of entering the loop, the first and second steps of the loop are executed. If the conditions for setting EX to "1" arise during the first step, a request to set EX to "1" is made by the second step (such as A10, F2, or R2), and exit from the loop occurs on the third step since ex = 1. If the conditions for setting EX to "1" do not arise during the first step in the loop, control continues to cycle through the loop until these conditions do arise on some odd-numbered step -- say 2k + 1. During the 2k + 2 step, EX is set to "1". Since ex = 1, exit from the loop occurs during the 2k + 3 step.

As shown on D-1289, the J memory element is conditionally set to "0" or "1" during G1 (D-1277) and is always set to "1" during V4 (D-1280). The j and $\overline{j}$ outputs feed the conditional logic associated with the G1, D1, R1, S1, S5, S9, V1 and V3 control areas. These signals condition the choice of R · MgM or R $\vee$ MgM during B2 as shown on RgM (D-1191). They also feed the store logic shown on HQR (D-1525) as discussed in section 3.1.2.14.

The N memory element appears on D-1290. It is conditionally set to "0" or "1" during G1 (D-1277) and is unchanged during other control steps. The n and $\overline{n}$ outputs influence the conditional logic in the D1, M1, R1, S1, S5, and S10 control areas. The $\overline{n}$ signal also feeds the store logic shown on HQR (D-1525) and is discussed in section 3.1.2.14.

Both OV and OV' appear on D-1291. They are used for overflow indication. The effect of the ODIR signal on the OV request input and the OV bypass logic is considered in section 5.1. Both OV and OV' are set to "0" when DC is cleared.

OV is conditionally set to "1" or bypassed during D7, D15, E3, F2, F5, K2, M3, and S9. It is conditionally set to "1" during D5 and conditionally bypassed during K1, K2, and M8. The ov output is used only by AC to detect accumulator overflow (both fractional and exponential). Only AC can set OV to "0" by causing Reset OV = 0.

The OV' memory element is conditionally set to "0" or "1" by the F1 (D-1274) control step only. It is used as a buffer storage element for OV during the inner loop of the shift, (F), sequence. Its outputs are used in F2 conditionally to bypass OV or set it to "1".

The RO memory element appears on D-1292. RO is set to "0" when DC is cleared. It is set to "1" whenever the fraction in A is to be rounded. In particular it is set to "1" during D1, M1, and V1. It is conditionally set to "0" or "1" during S1, and is always set to "0" during D3, M3, S9, and V3. The ro output conditions the round-off logic shown on ZDR (D-1505) and discussed in section 3.1.7.

The TC memory element also appears on D-1292. It is conditionally set to "0" or "1" by G1 (D-1277). It is always set to "0" during S11 (D-1284). Except for the CAE and CSE orders, TC is set to "1" for "store-clear" type instructions (STC, ASC, or SSC). The tc and $\overline{tc}$ outputs feed the conditional logic in the K2, S1, and S10 control areas.

As shown on D-1293, the X memory element is conditionally set to "0" or "1" by G1 (D-1277). It is always set to "1" during D4 (D-1271) and to "0"

during S6 (D-1283). The x and $\overline{x}$ outputs feed the conditional logic in the D1, D2, D3, S1, andS5 control areas. They also affect the logic between the low ends of R and FO as shown on LQR (D-1524) and discussed in section 3.1.2.14.

The Z memory element is shown on D-1294. It is set to "1" when the fraction in AQ becomes zero or when the exponent becomes less than -128. It is also set to "1" in D7 (D-1272) if ZA is set to "1" (i.e., za = 1) during (R) as as consequence of A containing zero. Z is set to "0" by any instruction which uses the (B) sequence and by the LAL instruction. The z and $\overline{z}$ outputs feed the conditional logic associated with the A1, A2, A3, D5, D6, F1, K1, K2, M1, M3, R1, S5, S7, and S9 control areas. The $\overline{z}$ output also goes to AC.

The ZA, ∇, and Δ_2 status memory elements appear on D-1295. All three of these serve very special purposes. Note that ∇ and Δ_2 are cleared to "0" by cdc.

ZA is always set to "0" during G1 (D-1276). It may be set either to "0" or "1" during R2 (D-1282). The latter setting occurs only when a single length normalization is being performed and A is found to contain zero. (A single length normalization is used only when R1 is entered from D4 of the divide sequence.) The za and $\overline{za}$ outputs feed the conditional logic in D5, D7, and R1.

The ∇ memory element is set to "1" during D13, S2, and also during S1 if tc = 1. It is set to "0" during D17, S11, and also during S1 if $\overline{tc}$ = 1. If $\overline{tc}$ = 1 and XCH is not being executed, ∇ is also set to "0" during S10. The ∂ and $\overline{\partial}$ outputs influence the formation of the σ_{-3} and σ_{-2} signals as discussed in section 3.1.2.6 and shown on HAS (D-1522). The $\overline{\partial}$ output is used to close the paths $\alpha_{43} \rightarrow R_{-1}$ and $\alpha_{44} \rightarrow R_0$ during S2 (D-1284). This latter use is mentioned in section 3.1.2.12 and is shown on HQR (D-1525).

The Δ_2 memory element is set to "1" by the request from D10 (D-1272) and reset to "0" by the request from D14 (D-1273). The (δ_2-a) output allows β_{42} to feed the input to R^*_{42} during the inner loop of the (D) sequence. This is discussed in section 3.1.17 and appears on LQR (D-1524). The (δ_2-b) output modifies the function performed by the carry-borrow logic (D-1527) during division

as discussed in section 3.1.13.

Θ_1 appears on D-1296. It is conditionally set to "0" or "1" during G1 (D-1277) and V3 (D-1280). It is always set to "1" during A6 (D-1268). The θ_1 and $\bar{\theta}_1$ outputs are inputs to the conditional logic associated with the D1, D2, D3, D5, M1, P1, R1, S10, and V4 control areas. They also feed the θ decoder as shown on $\mu\theta$D (D-1506) and discussed in section 3.1.14.

Θ_2 is shown on D-1297. It is conditionally set to "0" or "1" during G1 (D-1277). Its state is never changed during the execution of an instruction. The θ_2 and $\bar{\theta}_2$ outputs feed conditional logic in the B3, D1, D2, D3, D5, M1, P1, R1, S1, and S10 control areas. These signals also feed the θ-decoder as shown on $\mu\theta$D (D-1506), and condition the choice of R $\cdot$ MgM or R $\vee$ MgM during B2 as shown on RgM (D-1191).

The Λ memory element appears on D-1298. It is conditionally set to "0" or "1" during G1 (D-1277) and is not affected by other control steps. The λ and $\bar{\lambda}$ signals are inputs to the conditional logic in the B1, B2, B3, F1, F2, F4, F5, S1, S5, S7, and S9 control areas. The $\bar{\lambda}$ output also feeds the end connection logic shown on HAS (D-1522) and discussed in section 3.1.2.8.

The μ and Ω memory elements are shown on D-1299. These elements play very special roles. The former is used only during the (M) sequence while the latter is used only during the (A) and (F) sequences. They are both set to "0" when DC is cleared.

In M1 (D-1281) μ is always set to "1". It is likewise set to "1" during M6 until the final pass through the multiply loop during which μ is set to "0". The μ output controls the input paths to Q^*_{42} and R^*_{42} during the multiply loop (M5 and M6). This is discussed in section 3.1.15 and appears on LQR (D-1524).

The Ω memory element is conditionally set to "0" or "1" during A9 (D-1269) and F1 (D-1274). The Ω and $\bar{\Omega}$ outputs are inputs to the conditional logic of the A10 and F2 control areas as discussed in sections 5.5 and 5.8.

5. DELAYED CONTROL

Delayed Control (DC) obeys the orders it receives from Advanced Control (AC) by executing a partially ordered sequence of micro-operations in the Arithmetic Unit (AU)via the Link Mechanisms (LM). The AU and LM are discussed in the preceding sections. Each of the three types of micro-operations is associated with one of the three types of LM. The micro-operations are: the opening and closing of a register or memory element gate (such as gA), the setting of a selector mechanism (such as 4AQsSR), and the setting of a control status memory element (such as DL or $\overline{DL}$). A set of micro-operations is called a control step and is represented by a box on the DC Flow Chart. The time ordering of the micro-operations within a control step is not important so long as all memory elements (including selector mechanisms) which may change as a consequence of these operations are allowed to achieve their ultimate state before the active phase of the step is terminated. In particular, this means the F-element gate must remain on long enough for the input logic to settle and set the F-element.

An ordered sequence of control steps is called a control sequence. As shown on the DC Flow Chart, there are thirteen control sequences ranging from one control step for decode to twenty control steps for divide. The names and abbreviations of these sequences are listed below.

Floating Point Add - (A)

Clear Add - (B)

Divide - (D)

Exponent Arithmetic - - - - - - - - - - - - - - - - - - - (E)

Shift - (F)

Decode - (G)

Correct Overflow and Detect Zero - - - - - - - - - - - - (K)

Load Q - (L)

Multiply - (M)

Store Preliminaries - - - - - - - - - - - - - - - - - - (P)

Normalize - (R)

Store - (S)

Difference Absolute Value - - - - - - - - - - - - - - - (V)

Two or more control sequences are used during the execution of every DC order. Each order begins with the (G) sequence, continues with a series of one or more other sequences, and terminates with the next entry to (G). Except for the (E) sequence and the premature exits from (D) and (M), all order sequences must pass through one or more steps of the (K) sequence before terminating with an entry to (G). The (K) sequence detects and conditionally modifies the state of the fractional and exponent accumulator (AQ and E) before starting the next instruction. In particular, modulo 2 fractional overflow in AQ is detected and corrected by a right shift of one base 4 position. If the compensating exponent

correction causes e > 127, then OV is set to indicate accumulator overflow. If Z is not already set, the (K) sequence checks for a zero fraction in AQ. If so, it sets Z. The (E) sequence and the premature exits from (M) and (D) skip (K) and terminate in (G) directly. In each case the state of the accumulator is determined and reflected in the settings of OV and Z before the exit to (G) is made.

The control sequence series used to execute every DC order are listed below. In each series, the terminal (G) represents the decode step of the next order. The control step from which a premature exit occurs is indicated. A dotted line means that the exit is conditional while a solid line means it is unconditional. The three conditional exits from (K) to (G) are not indicated. No attempt is made to distinguish the various control paths through (S).

Table I: Control Sequence Series for DC Orders

Octal Code	Order Mnemonic	Sequence Series
00	CSB	
01	CST	
02	CAD	
03	CAT	(G) → (B) → (K) → (G)
04	NOT	
05	AND	
06	LOR	
07	BLS	
10	SUB	(G) → (A) → (K) → (G)
12	ADD	A1 ⇢ (B) ⇢ (K); Premature exit for case 1 addition
14	SSC	(G) → (A) → (K) → (R) → (S) → (K) → (G)
16	ASC	A1 ⇢ (B) ⇢ (K); Premature exit for case 1 addition

Table I: Control Sequence Series for DC Orders (Continued)

Octal Code	Order Mnemonic	Sequence Series
11	SBE	
13	ADE	(G) → (E) → (G)
15	CSE	
17	CAE	
20	MPY	(G) → (R) → (M) → (K) → (G) M8 Premature exit when multiplier is zero or the multiplicand exponent is -64
21	DIV	Return to (R) if the divisor is not normalized Premature exit when the divisor is zero D4 D5 (G) → (R) → (D) → (K) → (G) D8 Premature exit when the divisor is non-zero but the dividend is zero
22	NDV	Unconditional return to (R) with immediate bypass to D1 if the divisor is normalized D4 D5 (G) → (R) → (D) → (K) → (G) D8 See section 5.13

Table I: Control Sequence Series for DC Orders (Continued)

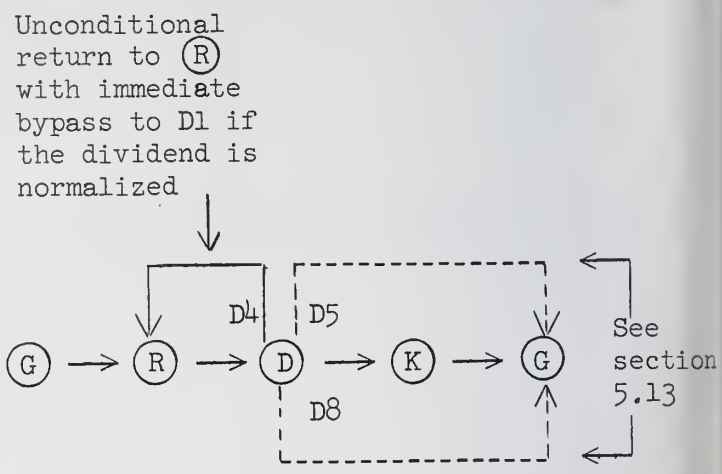
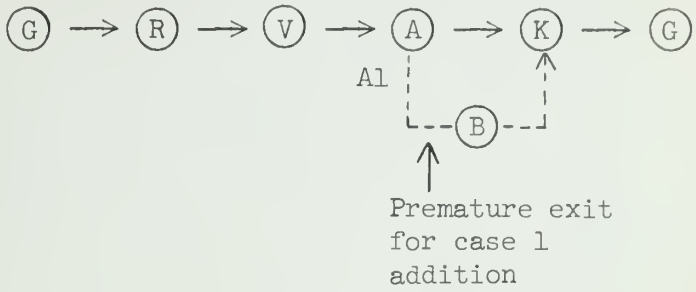
Octal Code	Order Mnemonic	Sequence Series
23	VID	
25	XCH	G → R → S → B → K → G
24	STR	G → R → S → K → G
26	STC	
27	STN	
30	STF	
31	SIF	G → P → F → S → K → G
32	SEQ	
33	SIA	
34	STU	
35	SAM	G → S → K → G
36	SAL	
37	SEX	
40	Illegal	
41	LAL	G → L → K → G

Table I: Control Sequence Series for DC Orders (Continued)

Octal Code	Order Mnemonic	Sequence Series
42	DAV	
43	SRM	G → S → K → G
44	Illegal	
46	Illegal	
45	LRS	
47	SRS	G → F → K → G

The logic of DC is based primarily on the concepts of asynchronous circuit theory as developed by D. E. Muller and W. S. Bartky.* As as consequence of economy and the desire for speed, the physical realization of DC does not fall within the class of speed-independent circuits unless some unrealistic assumptions are made concerning the generation and distribution of status signals such as link mechanism replies. For a discussion of logic used to realize DC and its departure from speed-independence, see Report 130, "Further Studies in Speed-Independent Logic for a Control" by R. E. Swartwout.

It is difficult to talk about the departure from speed-independence

*Report 75, "A Theory of Asynchronous Circuits I" by D. E. Muller and W. S. Bartky.

Report 78, "Theory of Asynchronous Circuits II" by D. E. Muller and W. S. Bartky.

in DC alone. A complete discussion must consider the design of the Arithmetic Subsystem as a whole. For example, there are not provisions in the Arithmetic Subsystem design for determining when the outputs of registers, adders, and decoders are reliable. Furthermore, the logic of DC does not check that the outputs of the status signal distribution systems have reached their correct state before the control step which they govern is initiated. To guarantee that such signals reach their correct state by the time they are needed in some future control step, the present control step and any intermediate steps must last longer than some minimum time. In DC the natural circuit delays and reply paths are usually adequate in this regard. In a few cases, artificial delays have been inserted in reply paths to insure that certain status signals are reliable before the next control step is initiated.

The logic drawings for the DC sequences are: (A) D-1268, D-1269; (B) left half of D-1270; (D) D-1271, D-1272, D-1273; (E) right half of D-1270; (F) D-1274 and right half of D-1275; (G) D-1276, D-1277, D-1278; (K) D-1279; (L) left half of D-1280; (M) D-1281; (P) left half of D-1275; (R) D-1282; (S) D-1283, D-1284; and (V) on the right half of D-1280.

The description of these sequences begins with (G) followed by (B) and (K) to give the reader an introduction to the logical operation of DC. Abbreviated descriptions of the remaining control sequences are then given in the following order: (L), (A), (E), (S), (F), (P), (R), (V), (M), and (D). In correlating the description with the DC Flow Chart (D-1128), the reader is warned that negative logic is used throughout DC but positive logic expressions appear on the chart.

A few comments are in order concerning certain symbols which appear on the logic drawings for DC. The cdc input to every control Eccles-Jordan represents the set signal which goes to "0" when the "Clear DC" button on the

console is pressed. The symbols (I) and (ID) represent drivers for indicator lights on the chassis and console.

The initial state of DC is established by pressing the "Clear DC" button on the console. This sets the MsA selector mechanism to KgA and all other selector mechanism to their 0-state. It clears CL, EXA, EXR, OV, OV', R0, Z, Δ_2, ∇, Ω, and μ to "0". It also sets all control Eccles-Jordans to "0" except D8 which is set to "1". The net effect is to produce an active request for G1 from D8, i.e., D8 - G1 = 0. G1 responds to this request as described below and begins to decode the order AC has placed in DCR.

5.1 The Decode Sequence (G)

The sequence consists of one control step, G1, with a "latch" as shown on G-I (D-1276). It uses and flushes the instruction decoder shown on G-II (D-1278) and the status memory element decoder (encoder) shown on G-III (D-1277). Using the order bits in DCR the instruction decoder selects and activates (sets to "1") the Eccles-Jordan of the initial control step in the sequence which follows (G). The status memory element decoder decides for each order the setting of certain status memory elements, whether to activate gEA, and how to set sEA and sD. The particular series of sequences which follow G depends primarily on how certain status memory elements are set. Note that G1 does not affect memory elements such as R0 which are turned on and off by the sequences which use them.

The following conditions are always true at the start of the decode (G1) step:

(1) The result of the previous order is held in AQ and E. For divide orders, the quotient is held in A and E while the remainder is held in R and ES.

(2) The fraction, f, in AQ is not overflowed, i.e. $-1 \leq f < 1$.

(3) KgA = 1 so the assimilated value of A appears as α at the output of the A-adder.

(4) The state of the MsS selector mechanism depends on the previous order or orders and corresponding operands.

(5) AC has previously placed the next order in DCR and set DCRFL to "1" to indicate DCR is "full." If the order placed in DCR does not require an initial operand, AC has set DCRS to "1". AC may or may not have loaded F1 (IN) and set INFL to "1" accordingly. If the order placed in DCR requires an initial operand, then AC must have set DCRS to "0", loaded F1 (IN), and set INFL to "1".

A summary of G1 requests as a function of the order in DCR is shown in Table II. A "0" or "1" indicates the requested state of a status memory element. A "1" also indicates a gate request while "0", "E", "EM", or "$\overline{\text{EM}}$" indicate the requested state of the sEA or sD selector mechanisms. If a check ($\surd$) appears, it means G1 does not request the corresponding link mechanism for the order indicated. In the case of EXR, an "e" appears whenever the normalize (R) sequence follows (G). The "e" indicates that EXR is set either to "0" or "1" according to the state of the accumulator. In positive logic, $\text{EXR} = z(d1) \vee (n\alpha) \vee (za)(\overline{d1}) \vee \bar{n}\,\bar{\theta}_1\bar{\theta}_2(QZ) \vee j(QZ)$. The last row of Table II shows the first active control step after G1.

It is important to note that F1gMEM and $\text{Og}Q_{43}Q_{44}$ are always turned on and off by G1. F1gMEM allows the contents of F1 (IN) to be transferred to M and EM in the MAU and EAU. $\text{Og}Q_{43}Q_{44}$ clears the two least significant bits of the Q register to "0". Furthermore, G1 always sets DCRFL, CR, and ZA to "0".

Table II: G1 Requests as a Function of the DC Order

Order	Octal Code	00	01	02	03	04	05	06	07	10	11	12	13	14	15
	Mnemonic	CSB	CST	CAD	CAT	NOT	AND	LOR	BLS	SUB	SBE	ADD	ADE	SSC	CSE
Gate and Selector States	FlgMEM	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	$OgQ_{43}Q_{44}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	gEA	√	√	√	√	√	√	√	√	1	1	1	1	1	1
	sEA	√	√	√	√	√	√	√	√	E	E	E	E	E	0
	sD	√	√	√	√	√	√	√	√	$\overline{EM}$	$\overline{EM}$	$\overline{EM}$	EM	$\overline{EM}$	$\overline{EM}$
Status Memory Element States	DCRFL	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	INFL	0	0	0	0	0	0	0	√	0	0	0	0	0	0
	C	√	√	√	√	√	√	√	√	0	1	0	0	0	1
	CR	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	DL	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	EXA	√	√	√	√	√	√	√	√	0	√	0	√	0	√
	EXF	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	EXR	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	J	0	0	0	0	1	0	1	0	0	0	0	0	0	0
	N	0	0	0	0	0	0	0	0	0	0	0	0	1	1
	TC	0	0	0	0	0	0	0	0	0	0	0	0	1	1
	X	0	0	0	0	0	0	0	1	0	0	0	0	0	0
	ZA	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Λ	0	0	0	0	0	1	1	1	0	0	0	0	0	0
	θ_1	0	0	1	1	0	1	1	1	0	0	1	1	0	1
	θ_2	0	1	0	1	0	0	0	1	0	1	0	1	0	0
Control Step Entry		← B1 →								A1	E1	A1	E1	A1	E1

Table II: G1 Requests as a Function of the DC Order (Continued)

Order	Octal Code	16	17	20	21	22	23	24	25	26	27	30	31	32	33
	Mnemonic	ASC	CAE	MPY	DIV	NDV	VID	STR	XCH	STC	STN	STF	SIF	SEQ	SIA
Gate and Selector States	FlgMEM	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	$OgQ_{43}Q_{44}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	gEA	1	1	√	√	√	√	√	√	√	√	1	1	1	1
	sEA	E	0	√	√	√	√	√	√	√	√	E	E	E	E
	sD	$\overline{EM}$	EM	√	√	√	√	√	√	√	√	0	0	0	0
Status Memory Element States	DCRFL	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	INFL	0	0	0	0	0	0	√	0	√	√	√	√	0	√
	C	0	0	√	√	√	√	√	√	√	√	0	0	0	0
	CR	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	DL	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	EXA	0	√	√	√	√	√	√	√	√	√	√	√	√	√
	EXF	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	EXR	√	√	e	e	e	e	e	e	e	e	√	√	√	√
	J	0	0	0	0	0	0	0	0	0	0	1	1	1	1
	N	1	1	0	0	0	0	1	1	1	1	0	0	0	0
	TC	1	1	0	0	0	0	0	0	1	0	0	0	0	0
	X	0	1	0	0	0	0	0	0	0	1	0	0	0	0
	ZA	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Λ	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	θ_1	1	1	0	0	1	1	0	1	1	1	0	0	1	1
	θ_2	0	1	0	1	0	1	0	0	0	1	0	1	0	1
Control Step Entry		A1	E1	←				R1			→	←		P1	→

Table II: G1 Requests as a Function of the DC Order (Continued)

Order	Octal Code	34	35	36	37	40	41	42	43	44	45	46	47
	Mnemonic	STU	SAM	SAL	SEX	Illegal	LAL	DAV	SRM	Illegal	LRS	Illegal	SRS
Gate and Selector States	FlgMEM	1	1	1	1	1	1	1	1	1	1	1	1
	$OgQ_{43}Q_{44}$	1	1	1	1	1	1	1	1	1	1	1	1
	gEA	√	√	1	1	√	√	√	√	√	1	√	1
	sEA	√	√	E	E	√	√	√	√	√	0	√	0
	sD	√	√	0	0	√	√	√	√	√	EM	√	EM
Status Memory Element States	DCRFL	0	0	0	0	0	0	0	0	0	0	0	0
	INFL	√	√	√	√	?	0	0	√	?	0	?	0
	C	√	√	0	0	√	√	√	√	√	0	√	0
	CR	0	0	0	0	0	0	0	0	0	0	0	0
	DL	1	1	1	1	1	1	1	1	1	1	1	0
	EXA	√	√	√	√	√	√	√	√	√	√	√	√
	EXF	√	√	√	√	√	√	√	√	√	0	√	0
	EXR	√	√	√	√	√	√	e	√	√	√	√	√
	J	0	0	0	0	1	1	1	1	1	1	1	1
	N	0	0	0	0	0	0	0	0	0	0	0	0
	TC	0	0	0	0	0	0	0	0	0	0	0	0
	X	0	0	0	1	0	0	0	0	0	0	0	1
	ZA	0	0	0	0	0	0	0	0	0	0	0	0
	Λ	0	1	1	1	0	0	0	0	0	1	1	1
	θ_1	0	1	1	1	0	0	1	1	0	1	1	1
	θ_2	0	0	0	1	0	1	0	1	0	0	0	1
Control Step Entry		S1	S1	S2	S2	√	L1	R1	S3	√	F1	√	F1

The settings of C, sEA, sD, and the use of gEA are made according to the desired output of the exponent adder when the first control step following G1 is initiated. The accumulator exponent is in E while the exponent from memory is placed in EM by FlgMEM. EsEA and gEA place the accumulator exponent in EA while OsEA and gEA clear EA to zero. The output of the exponent adder (EAD) is then determined solely by the setting of sD and C as discussed in section 3.2.3.

INFL is set to "0" if the order being decoded requires an initial operand; otherwise, G1 makes no attempt to set it. EXA, EXF, and EXR are set according to whether the (A), (F), or (R) sequences follow (G). The set pattern for J has no clear description. (J) is set to "1" for NOT, LOR, LAL, DAV, SRM, and all orders which use the (F) sequence. (N) is set to "1" for all store type instructions which use the (R) sequence as well as for CSE and CAE. TC is set to "1" for store clear type instructions and also CSE and CAE. X is set to "1" for all orders whose least significant octal digit is 7. Λ is set to "1" for all logical type orders. θ_1 and θ_2 are set to control the choice of MsS and MsA settings in the (A) and (B) sequences. They are also used to control conditional paths in the (B), (D), (M), (P), (S), and (V) sequences.

To gain a better understanding of the decode logic, we consider the signal flow on G-I (D-1276).

In the inactive state, the G1 Eccles-Jordan shown on D-1276 is set to "0" (i.e., its "1" side has a "0" or negative voltage output). Consequently, the $\overline{A}^*$ has a "1" output which represents an inactive request. Note also that the ODIR output is "0". The last control step in the previous order must be K1, K2, D5, D8, E3, or M8 as indicated by the inputs to the restoring AND in the center of the drawing. Each of these final steps request some of the link mechanisms whose replies are inputs to the G1 $\overline{A}^*$. However, all of these final steps set OV to "1" or bypass it to cause OVr to go from "1" to "0". As shown on D-1291,

these requests are effective only when ODIR = 0. When OVr and the output of the restoring AND on D-1276 both go to "0", the G1 Eccles-Jordan is set to "1", but the output of the associated $\overline{A}^*$ is held to "1". ODIR now goes to "1", indicating that "the preceding order is done and all indicators are ready for inspection by AC." If a JOV order is present in AC, #5-g$\overline{OV}$ is allowed to go to "0" (shown on D-1291) and set OV to "0". OVr goes from "0" to "1" when #5-g$\overline{OV}$ goes back to "1". If a JOV order is not present in AC, #5-g$\overline{OV}$ remains at "1" and ODIR = 1 causes OVr to go from "0" to "1", even though the request from the final control step of the previous order is still active (i.e., still "0").

At the same time, the fact that the G1 Eccles-Jordan is now set to "1" permits the Eccles-Jordan associated with the active control step preceding G1 to be set to "0". This does not actually occur until all the requested link mechanism other than OV have responded with "0" replies and dec = [(DCRFL)d][(INFL)d $\vee$ dcrs] = 1. The latter signal is "1" when DCRFL is set to "1" and either INFL or DCRS is set to "1", indicating that the next order is in DCR and its operand - if needed - is in F1 (IN).

When the Eccles-Jordan in the control step preceding G1 is set to "0", the output of its associated $\overline{A}^*$ goes from "0" to "1". This causes the replies to go from "0" to "1", indicating that the activated link mechanisms are either off (in the case of gates) or latched in the state requested. When the OVr also goes to "1" as described above, the output of the G1 $\overline{A}^*$ goes from "1" to "0", initiating the active phase of decode. The requests made by this output depend on the order in DCR as summarized in Table II.

The "0" output of the G1 $\overline{A}^*$ causes the contents of F1 (IN) to be gated into M and EM, regardless of whether the order being decoded will use this operand. It also clears Q_{43} and Q_{44} to "0", and sets CR and ZA to "0" (i.e., requests $\overline{CR}$ and $\overline{ZA}$). It also allows the outputs of the DCR register to be decoded as shown on G-II (D-1278). Before the output of the G1 $\overline{A}^*$ goes to "0", all points in both

the instruction decoder (D-1278) and status memory decoder (encloder) are held to "1". The logic of these decoders does not react to the new outputs of DCR until G1-b goes to "0". The G-II (D-1278) logic selects the entry control step of the sequence which follows decode while the G-III (D-1277) logic sets the status memory elements to their appropriate state.

The negative logic for G-II is given below. The numbers in brackets indicate the range of octal digit pairs for which the associated signal is "0". The f_i and $\bar{f}_i$ signals represent the true and complement outputs of the $i^{\underline{th}}$ F-element in the DCR.

First Level

$$G10 = (G1 - b) \vee f_1 \vee f_2 = [00\text{-}17]$$

$$GX - a = GX - b = G12 = (G1 - b) \vee f_1 \vee \bar{f}_2 \vee f_3 = [20\text{-}27]$$

$$GY = G13 = (G1 - b) \vee f_1 \vee \bar{f}_2 \vee \bar{f}_3 = [30\text{-}37]$$

$$GZ = G17 = (G1 - b) \vee \bar{f}_1 \vee f_2 \vee f_3 = [40\text{-}47]$$

Second Level

$$G1 - B1 = GB - a = GB - b = G10 \vee f_3 = [00\text{-}07]$$

$$GW = G11 = G10 \vee f_3 = [10\text{-}17]$$

$$G1 - P1 = GP = G21 = G13 \vee f_4 = [30\text{-}33]$$

$$G14 = G13 \vee \bar{f}_4 = [34\text{-}37]$$

$$G16 = G17 \vee f_4 \vee \bar{f}_5 \vee f_6 = [42]$$

$$G18 = G17 \vee \bar{f}_6 = [41, 43, 45, 47]$$

Third Level

$$G1 - A1 = GA = G11 \vee f_6 = [10, 12, 14, 16]$$

$$G1 - E1 = GE = G11 \vee \bar{f}_6 = [11, 13, 15, 17]$$

$$G1 - S2 = G2S = G14 \vee \bar{f}_5 = [36, 37]$$

$$G1 - S1 = G15 = G14 \vee f_5 = [34, 35]$$

$$G1 - R1 = (G12)(G16) = [20\text{-}27, 42]$$

$$G1 - L1 = G19 = G18 \vee f_4 \vee f_5 = [41]$$

$$G1 - S3 = G20 = G18 \vee f_4 \vee \bar{f}_5 = [43]$$

$$G1 - F1 = GF = G18 \vee \bar{f}_4 = [45\text{-}47]$$

The negative logic for G-III is given below. The numbers in brackets have the same meaning as above. An "x" indicates that any octal digit applies. Note that some of the variables in this logic are generated above, e. g., GY.

First Level

$$G30 = G31 = G32 = G33 = (G1 - f) \vee f_4 = [x0\text{-}x3]$$

$$G34 = (G1 - a) \vee \bar{f}_5 \vee \bar{f}_6 = [x3, x7]$$

$$G35 = (G1 - f) \vee f_6 = [x0, x2, x4, x6]$$

$$G36 = (G1 - a) \vee \bar{f}_4 \vee \bar{f}_6 = [x5, x7]$$

$$G37 = (G1 - f) \vee f_5 \vee f_6 = [x0, x4]$$

$$G38 = (G1 - f) \vee f_4 \vee \bar{f}_5 = [x2, x3]$$

$$G39 = (G1 - a) \vee \bar{f}_4 \vee \bar{f}_5 \vee f_6 = [x6]$$

$$G40 = G41 = (G1 - a) \vee \bar{f}_4 = [x4\text{-}x7]$$

Second Level

$G1 - C = G42 = GE \vee f_5 = [11, 15]$

$G43 = GE \vee \overline{f}_5 = [13, 17]$

$G1 - X = G44 = G34 \vee \overline{f}_4 = [x7]$

$G45 = (GB - a) \vee G36 = [05, 07]$

$G46 = GW \vee G41 = [14\text{-}17]$

$G47 = (GX - a) \vee G36 = [25, 27]$

$G48 = (GX - b) \vee GW = [10\text{-}27]$

$G49 = G40 \vee f_5 = [x4, x5]$

$G - 50 = GY \vee \overline{f}_4 = [34\text{-}37]$

$G - 51 = G36 \vee G39 = [x5, x6, x7]$

$G - 52 = (GB - b) \vee GZ = [00\text{-}07, 40\text{-}47]$

Third Level

$G1 - \overline{DL} = G53 = G44 \vee GZ = [47]$

$G54 = (GB\text{-}b) \vee G35 \vee G41 = [04, 06]$

$G55 = (GB - a) \vee G31 = [00\text{-}03]$

$G56 = (GX - b) \vee G41 = [24\text{-}27]$

$G57 = (GY)(G52) = [00\text{-}07, 30\text{-}37, 40\text{-}47]$

$G58 = (GX - a) \vee G39 = [26]$

$G59 = G33 \vee G48 = [10\text{-}13, 20\text{-}23]$

$G60 = (GX - b) \vee G37 = [20, 24]$

$G61 = G39 \vee G49 = [x4, x5, x6]$

$G62 = G31 \vee f_5 = [x0, x1]$

$G63 = G31 \vee \overline{f}_6 = [x1, x3]$

$G64 = G50 \vee G51 = [35\text{-}37]$

$G65 = G51 \vee G52 = [05, 07, 45\text{-}47]$

$G66 = GF \vee GP \vee G2S = [30\text{-}33, 36, 37, 45, 47]$

$G67 = GE \vee G40 = [15, 17]$

$G68 = GE \vee G33 = [11, 13]$

Fourth Level

$G1 - \overline{C} = (GA)(G43)(G66) = [10, 12\text{-}14, 16, 17, 30\text{-}33, 36, 37, 45, 47]$

$G1 - DL = (GB\text{-}b)(GY)(G32)(G48)(G61) = [00\text{-}46]$

$G1 - J = (GP)(GZ)(G54) = [04, 06, 30\text{-}33, 40\text{-}47]$

$G1 - \overline{J} = (GX - a)(GW)(G45)(G50)(G55) = [00\text{-}03, 05, 07, 10\text{-}27, 34\text{-}37]$

$G1 - N = (G46)(G56) = [14\text{-}17, 24\text{-}27]$

$G1 - \overline{N} = (G57)(G59) = [00\text{-}13, 20\text{-}23, 30\text{-}47]$

$G1 - TC = (G46)(G58) = [14\text{-}17, 26]$

$G1 - \overline{TC} = (G47)(G57)(G59)(G60) = [00\text{-}13, 20\text{-}25, 27\text{-}47]$

$G1 - \overline{X} = (G32)(G61) = [x0\text{-}x6]$

$G1 - \theta_1 = (G38)(G51) = [x2, x3, x5\text{-}x7]$

$G1 - \overline{\theta}_1 = (G37)(G62) = [x0, x1, x4]$

$G1 - \theta_2 = (G34)(G63) = [x1, x3, x7]$

G1 - $\overline{\theta}_2$ = (G35)(G61) = [x0, x2, x4-x6]

G1 - Λ = (G64)(G65) = [05-07, 35-37, 45-47]

G1 - $\overline{\Lambda}$ = (G32)(G37)(G48) = [00-04, 10-34, 40-44]

G1 - gEA = (GW)(G66) = [10-17, 30-33, 36, 37, 45, 47]

G1 - OsEA = (GF)(G67) = [15, 17, 45, 47]

G1 - EsEA = (GA)(GP)(G2S)(G68) = [10-14, 16, 30-33, 36, 37]

G1 - OsD = (GP)(G2S) = [30-33, 36, 37]

G1 - EMsD = (GF)(G43) = [13, 17, 45, 47]

G1 - $\overline{\text{EM}}$sD = (GA)(G42) = [10-12, 14-16]

For example, if a clear add type order is in DCR, then the G1 - B1 output of the G-II logic goes from "1" to "0", causing the B1 Eccles-Jordan shown on D-1270 to set to "1" but holds the output of the G1 $\overline{A}^*$ to "1". The resulting "0" output on the "0" side of the G1 Eccles-Jordan is OR'd with G1 - B1 to yield a "0" which is returned to the AND in G1 shown at the bottom left corner of G-I (D-1276). The output of this AND is OR'd with the output of the larger OR which collects the replies from the status memory elements that are set by the outputs of the G-III logic as well as FlgMEMr, (DCRFL)d, and (INFL)d. A discussion of the logic associated with the latter two signals is postponed for the present. Note that gEA, sEA, sD, and C are not requested for (B) type orders, so their replies remain at "1". It is clear that when all of the activated replies are "0", the top input to the A|$\overline{A}$ will go to "0", causing the AND output to set the G1 Eccles-Jordan back to "0". This in turn changes the output of the G1 $\overline{A}^*$ from "0" to "1" which initiates the relaxation phase of the decode control step.

If DCR contained an illegal order (40, 44, or 46), the G-II logic would not set the entry Eccles-Jordan of any control sequence. As a result, both

inputs to the $A|\overline{A}$ shown on G-I (D-1276) would remain at "1" so that the $\overline{AND}$ output would remain at "0". This "0" OR'd with the "0" request from the G1 $\overline{A}^*$ would cause the "decode error" indicator to light.

So far, nothing has been said about the output of the "latch" AND that appears at the right edge of G-I (D-1276). Its outputs, G2 - a, b, d, and k, go to "0" after the G1 $\overline{A}^*$ output goes to "0" during the active phase of decode.

G2 - a, G2 - b, and G2 - d "unlatch" the OR's shown at the bottom of G-I. The second inputs to these OR's are the replies from those link mechanisms which are requested by other control steps in DC. The outputs of these OR's are denoted by primes and feed only the logic shown on G-I. Since FlgMEM, N, Λ, and θ_2 are requested only during decode, their replies feed the logic on G-I directly. The significance of this will soon be apparent.

The "0" G2 - d output is also distributed to the entry $\overline{A}^*$ of all control sequence which may follow G1. For example, G2 - f is an input to the B1 $\overline{A}^*$ shown on D-1270. G2 - f = 0 holds the output of the $\overline{A}^*$ at "1" even after the G1 - B1 request goes back to "1".

The "0" G2 - k output sets DCRFL to "0" which returns the "dec" signal to "0" and causes (DCRFL)d = 0. If dcrs = 0, it also sets INFL to "0" which in turn causes (INFL)d to change from "1" to "0". If dcrs = 1, (INFL)d is already "0". The $\overline{(DCRFL)d}$ signal remains at "1" during this time and continues to inhibit AC from reloading DCR and setting DCRFL back to "1". The same is true of $\overline{(INFL)d}$ if INFL is set to "0" by G2 - k.

After the G1 Eccles-Jordan is set to "0" and the $\overline{A}^*$ output changes from "0" to "1", all nodes and outputs of the G-II and G-III logic must return to "1". In fact, R. R. Shively designed this logic such that its outputs are "1" only after internal nodes have returned to "1". This logic is said to be

"flushed" to "1" when the replies from the control step entries, gEA, sEA, sD, C, and all other status memory elements have returned to "1". For any given order, only one control step entry reply (e.g., B1 - G1) goes to "0" so only one must go back to "1". To check this RP - 1 and RP - 2 on G-I feed the "latch" AND. RP - 3 and RP - 4 also feed the "latch" AND to insure that a "0" is not trapped between the series OR's which gather replies. The need for such a check when reply OR's are used in series was pointed out by R. E. Swartwout.

When all replies have returned to "1", the output of the "latch" AND changes from "0" to "1". The latching effect is now clear. The primed reply signals on G-I cannot change from "1" to "0" even though the unprimed replies will change in response to requests from control steps which follow G1. The fact that FlgMEM, N, Λ, and θ_2 are only requested by G1 means that the "latch" AND output remains at "1" until the G1 $\overline{A}^*$ output goes to "0" during the next decode step.

When G2 - k goes to "1", $\overline{(DCRFL)d}$ and possibly $\overline{(INFL)d}$ go from "1" to "0". This allows AC to load DCR with a new order and set DCRFL back to "1". The same applies to F1 (IN) and INFL when necessary.

When G2 - d goes to "1", the corresponding cable inputs to the entry $\overline{A}^*$s also go to "1". For example, in the (B) sequence (D-1270), the G2 - f signal goes to "1". If the B1 Eccles-Jordan was set to "1" by G1 - B1 during the active phase of decode, the output of the B1 $\overline{A}^*$ changes from "1" to "0", initiating the first step (B1) of the (B) sequence. The decode entry to other sequences follows a similar pattern.

5.2 The Clear Add Sequence (B)

Besides the decode entry to B1, there are also entries from A1 and S10 as shown on D-1270. The entry from A1 corresponds to the case where the addend (subtrahend) is so much larger than the augend (minuend) in AQ that a

CAD (CSB) yields the sum (difference). The entry from S10 has CAD initial conditions and represents the second half of an XCH order.

Let m and em denote the contents of M and EM as usual. Remember that $m_{-1} = m_0$ amd $em_7 = em_6$ following G1 since the sign bits of the fraction and exponent are duplicated as they are placed in M and EM via FlgMEM. Λ is set to "0" for CSB, CST, CAD, CAT, and NOT. These orders place -m, -2m, m, 2m, and $\bar{m}$ in A and transfer em to E. Since $\bar{m}$ represents the bitwise complement of m, $-m = \bar{m} + 2^{-44}$ and $-2m = \overline{2m} + 2^{-44}$. For AND, LOR, and BLS, Λ is set to "1". These orders form $a \cdot m$, $a \vee m$, and 2a with the sign bit duplicated. The result is placed in A while the contents of E remain unchanged.

The (B) sequence sets Z to "0" (i.e., sets $\bar{Z}$) and sets KgA = 1 in all cases. The output of the A-adder, α, represents the assimilated value of the accumulator, a, at the time of exit to K1. Because of the CSB, CST, and CAT orders, α may be overflowed modulo 2 ($\alpha < -1$ or $\alpha \geq 1$) when control is transferred from B3 to K1. This condition is detected and corrected by the (K) sequence as discussed in section 5.3.

To acquaint the reader with the logic of DC, we follow the signal flow through the (B) sequence as shown on D-1270.

Assume that the output of the B1 $\overline{A^*}$ has just changed from "1" to "0" following an entry from G1, A1, or S10. This "0" request from B1 sets the sSR selector mechanism to OAsSR and causes gS = gR = 1 in the MAU. KgA = 1 at the time of entry to G1, so it is still true at B1. The same is true for the entry from A1 (D-1268). The MsA selector mechanism is set to KgA in S1 (D-1283) and never reset during (S), so KgA = 1 when B1 is entered from S10. Consequently, α represents the assimilated value of A which is transferred to R when gR = 1. At the same time, zero is placed in S when gS = 1. Z is set to "0" since the quantity which is eventually placed in AQ is generally not zero.

If a NOT order is being executed, $\bar{j} = \lambda = 0$ so that an $\overline{\text{MsS}}$, $\overline{\text{CS}}$ request is sent to the MsS selector mechanism. This causes $\bar{m}$ to be added to the contents of S without a complement carry in the $44^{\underline{\text{th}}}$ position. If any other (B) type order is being executed, $j \cdot \bar{\lambda} = 0$ so that the θMsS decoder shown on D-1506 is enabled. The settings of θ_1 and θ_2 determine the state requested. Note that AND and LOR request MsS while BLS requests 2MsS. Since S contains zero in all cases, the state of the MsS mechanism and the contents of M determine σ which is the output of the S-adder.

In the EAU, the "0" B1 request sets the sEA mechanism to the OsEA state and opens gEA, thereby causing EA to be cleared to 0. C is set to "0" and the sD mechanism is set to the EMsD state. This causes the contents of EM to appear as the output, d, of the exponent adder.

While all of this is taking place, B1 - B2 is "0" and thus sets the B2 Eccles-Jordan to "1", but at the same time prevents the output of the associated $\overline{A^*}$ from going to "0". When the false side (i.e., "0" side) of the B2 Eccles-Jordan goes to "0", it is set. This is checked by OR'ing it with the "0"-going replies from the activated link mechanisms. When all are "0", the B1 Eccles-Jordan is returned to "0". This is checked by the presence of a "0" on the true output of the B1 Eccles-Jordan. This "0" causes the link mechanism replies to change from "0" to "1" which completes the relaxation phase of B1 and initiates the active phase of B2.

All entries to (B) for which $\lambda = 0$ bypass B2 and make the requests associated with B3, but do not set the B3 Eccles-Jordan. This is accomplished via a restoring AND following the B3 $\overline{A^*}$. The requests made by this AND are considered later.

Λ is set to "1" for the AND, LOR, and BLS orders, so the B2 step is done. The B2 - RgM request goes from "1" to "0", thus enabling the conditional

logic shown on RgM (D-1191). For the AND order, $j \vee \overline{\lambda} = 0$ and $j \vee \theta_2 \vee \overline{\lambda} = 0$, so R · MgM is done while R ∨ MgM is bypassed to obtain a "0" reply. Since the assimilated contents of A were placed in R during B1, R · MgM produces a bitwise AND of the assimilated contents of A with the contents of M. The result replaces the contents M as discussed in section 3.1.2.2. For the LOR order, $\overline{\lambda} \vee \overline{j} \cdot \overline{\theta}_2 = 0$ and $\overline{j} \vee \overline{\lambda} = 0$ so that R ∨ MgM is done and R · MgM is bypassed. This places the bitwise OR of the assimilated contents of A and M in M. For the BLS order, $j \vee \overline{\lambda} = 0$ and $\overline{\lambda} \vee \overline{j}\ \overline{\theta}_2 = 0$ so RgM is done as discussed in section 3.1.2.2. Consequently, the assimilated contents of A -- now held in R -- are transferred to M. For all three orders, σ must change to match the new contents of M.

Meanwhile, B2-B3 sets the B3 Eccles-Jordan to "1" but prevents the output of the corresponding $\overline{A}^*$ from going to "0". When the R ∨ MgM and R · MgM replies go to "0", the nonrestoring AND output follows and sets the B2 Eccles-Jordan to "0". This in turn causes the output of the B2 $\overline{A}^*$ to go from "0" to "1", thereby initiating the relaxation phase of B2. When the R · MgM and R ∨ MgM replies change from "0" to "1", the B3 $\overline{A}^*$ output goes from "1" to "0", initiating the active phase of B3.

Note that the output of the B3 $\overline{A}^*$ feeds a restoring AND which was mentioned above in regard to bypassing B2. When the output of this AND changes from "1" to "0", either the Eccles-Jordan in B3 is set to "0" and the one in B2 is still set to "1", or the Eccles-Jordan in B3 is set to "1" and the one in B2 is set to "0". In either case, the "0" request from the AND in B3 sets the sAQ mechanism to SOsAQ and opens gA so that σ, which equals the contents of M modified by the state of the MsS selector mechanism, is transferred straight into A. The MsA mechanism is set to KgA. After the carry generator and A-adder have responded, α represents the assimilated value of σ as placed in A. At this

point, the stored carries in A are all "0" except for a possible carry in A^*_{42} when CSB and CST are executed.

For the so-called arithmetic orders (CSB, CST, CAD, CAT, and NOT) $\lambda = 0$ so the B3 request opens gQ and DgE. This clears Q to zero and places the exponent of the incoming operand in E (i.e., the quantity that was placed in EM during G1 is now transferred to E). For the logical orders (AND, LOR, and BLS), $\lambda = 1$ so gQ and DgE are bypassed. This means that the contents of Q and E remain as they were during G1.

Except for BLS, B3 also requests entry to K1. The B3-K1 signal is "0" when B3 is active and $\lambda\theta_2 = 0$. This signal sets the K1 Eccles-Jordan to "1" and causes K1 - f to go to "0" as soon as all the link mechanisms requested by B3 respond with "0" replies. The K1 logic is shown on D-1279. If BLS is the order being executed, $\bar{\lambda} \vee \bar{\theta}_2 = 0$ so the B3-K1 signal remains at "1" and the B4 Eccles-Jordan is set to "1" by the "0" B3-B4 signal. The B4-B3 signal goes to "0" as soon as all the link mechanisms requested by B3 respond with "0" replies.

As soon as K1 - f or B4-B3 go to "0", the Eccles-Jordans in B2 and B3 are set to "0". One of these is already set to "0" depending on whether or not the B3 Eccles-Jordan was set to "1" by a request from B2. In either case, the result is the same. The output of the AND in B3 changes from "0" to "1". When the link mechanisms respond with "1" replies, either K1 or B4 becomes active.

Assume for the present that BLS is being executed so B4 becomes active. Since the assimilated contents of A during G1 are now shifted left by one bit position, it is possible for $a_{-1} \neq a_0$. There are no stored carries in A so $\alpha_{-1} = a_{-1}$ and $\alpha_0 = a_0$. If $\alpha_{-1} \neq \alpha_0$, the K1 step will shift the contents of AQ one base 4 position to the right and yield a binary right shift of the original contents of A. To prevent this, the "0" request from the B4 $\bar{A}^*$ opens $a_0 gA_{-1}$ ($a_0 gA_{-1} = 0$ in this case) which copies a_0 into A_{-1}. As a result, $\alpha_{-1} = \alpha_0$ at the time of entry to K1 from B4. Meanwhile, the B4-K1 signal sets

the K1 Eccles-Jordan (shown on D-1279) to "1". There is no reply associated with $a_0 gA_{-1}$ so the "0" output of the K1 Eccles-Jordan is sufficient to cause K1 - d to change from "1" to "0". This signal sets the B4 Eccles-Jordan back to "0", which causes B4 - $a_0 gA_{-1}$ and B4-K1 to return to "1". This initiates the active phase of K1 as described in the next section.

5.3 The Correct Overflow and Detect Zero Sequence (K)

Except for SBE, ADE, CSE, CAE, and premature exits from MPY, DIV, NDV, and VID orders, all orders pass through one or more control steps of the (K) sequence. In all cases, KgA = 1 at the time of entry to K1, so α represents the assimilated value of A. K1 determines whether AQ is overflowed modulo 2 or zero. These conditions cannot both be true. If examination of the most significant bits of α and the state of Z shows that neither of these conditions are true, K1 sends a bypass request to the OV status memory element and exits to G1 via K1-G1. If α is overflowed, K1 and K2 right shift AQ one base 4 position and check for exponent overflow before exiting to G1 or K3. If α is not overflowed, Z is set to "0", and the most significant bits of α are zero, K1 and K2 right shift AQ into SR and check for zero but do not return this assimilated representation to AQ. In K2, OV is bypassed and Z is set to "1" or bypassed, depending on whether SR is zero or not. Control then passes to G1 or K3 depending on whether TC is set to "0" or "1".

When SSC or ASC are executed, the (K) sequence is used twice as shown in Table I. On the first pass through (K) following (A), TC is set to "1" so K3 may be entered from either K1 or K2. K3 acts as an abbreviated decode step in that it sets CR to "0", clears Q_{43} and Q_{44} to "0", and opens the gate to EXR in the process of activating R1 (D-1282).

As a final example of signal flow through control logic, we consider the (K) sequence in detail.

The Eccles-Jordan in K1 may be set to "1" by an entry request from any one of eleven different control steps as shown on D-1279. Assume the replies associated with the entry request have gone to "0" and are now returning to "1". As soon as all reply inputs to the K1 $\overline{A}^*$ are "1", its output changes from "1" to "0", initiating the active phase of the K1 control step.

If $(\alpha ov) \vee \overline{z}(\overline{\alpha nz}) = 0$, the contents of AQ are neither overflowed modulo 2 nor zero. The significance of αov and αnz is discussed in section 3.1.9. Their values are valid at this point, since KgA = 1. If tc = 0, the K1-G1 signal requests entry to G1 and bypasses OV to obtain a "0" OVr as discussed in section 5.1. If the conditions for starting a new decode step are satisfied (i.e., if dec = 1), G1-K1 changes from "1" to "0" which sets the K1 Eccles-Jordan back to "0", beginning the relaxation phase of the K1 step. If, on the other hand, tc = 1, an entry request is sent to K3. The K2-K3 signal goes from "1" to "0" and sets the K3 Eccles Jordan to "1" which in turn causes K2-K1 to set the K1 Eccles Jordan back to "0". Therefore, when the output of the K1 $\overline{A}^*$ changes from "0" to "1" in the relaxation mode, the G1 or K3 control step becomes active depending on the setting of TC.

If $(\overline{\alpha ov})(z \vee (\alpha nz)) = 0$, the contents of AQ are either overflowed or Z is set to "0" and α may be zero. If $\alpha ov = \alpha_{-1} \oplus \alpha_0 = 1$, α -- and hence AQ -- is overflowed modulo 2. If $\alpha nz = \alpha_0 \vee \alpha_1 \vee \alpha_2 \vee \alpha_3 \vee \alpha_4 = 0$, α may be zero. Since z = 0, it is necessary to check every bit of α and Q to determine if all are zero. This is done by examining the outputs of the S and R zero detectors following a right shift into SR.

When the output of the AND-OR defined by $K1 \vee (\overline{\alpha ov})(z) \vee (\alpha ov)(\alpha nz)$ is "0", it sets the sSR mechanism to the 1/4 AQsSR state and opens gS. It also opens gR if DL is set to "1", indicating a double length operation. If DL is set to "0", gR is bypassed to obtain gRr = 0. In the latter case, α_{43}

and α_{44} are lost. DL is always set to "1" except when K1 is entered from D20, in which case dl = 0 to prevent losing the remainder which is stored in R.

In case 3 addition, CR is set to "1" during a particular pass through A10 in the (A) sequence, but it is not set to "0" prior to exit from (A). Thus CR is always set to "0" during K1 to insure that the true sign bit of the unassimilated number in AQ is copied into S_{-1} and S_0 when $\lambda = 0$ as discussed in section 3.1.2.8. If $\lambda = 1$, setting CR to "0" insures that "0" is inserted in S_{-1}, S_0, and S_1 when gS = 1. This is in keeping with the fact that α_0 is considered the left-most bit at the output of the A-adder when "logical" orders are executed. In K1, α_0 is transferred to S_2 when gS = 1. Since KgA = 1, all of the α_1^* bits are "0".

The "0" request from K1 also sets the MsS mechanism to the OMsS state. This causes the new contents of S to appear as σ at the output of the S-adder.

Additional activity occurs in the EAU during K1. The "0" request sets EsEA and opens gEA. The exponent, e, associated with AQ is thus transferred from E to EA. Since this request also sets OsD, the output of the exponent adder is therefore determined by the contents of EA and the setting of C. If $\alpha ov = 1$, C is set to "1", thereby adding a unit to the exponent in EA. This compensates arithmetically for the right shift of AQ to correct the fractional overflow condition. The output, d, of the exponent adder (D-1503) is e + 1 which may exceed 127. If so, dov = 1 at the output of the exponent decoder (D-1504) as discussed in section 3.2.4. Otherwise dov = 0. If $\alpha ov = 0$, C is set to "0" since AQ is left as it is, except for a transfer into SR to check for zero. In this case, d = ea but is not used since the non-overflowed branch of K2 does not open DgE.

While the above requests are made, K1-K2a sets the K2 Eccles-Jordan to "1" but prevents the output of the associated $\overline{A}^*$ from going to "0". When

the replies from the activated link mechanism change from "1" to "0", K2-K1 goes from "1" to "0" and sets the K1 Eccles-Jordan to "0". At the completion of the relaxation phase, all replies return to "1", including the K1-K2a request. The output of the K2 $\overline{A^*}$ then goes from "1" to "0".

The K2 step has a conditional branch in its control path. If c = 1, αov was true in K1, so SR is transferred straight into AQ, d is gated into E, and OV is set to "1" if dov = 1. If c = 0, αov was not true in K1, so SR is checked for zero and OV is bypassed to cause OVr = 0 when entering G1.

First consider the case where c = 1. On D-1279 $\overline{c}$ = 0, so K2 requests gA and sets SRsAQ which transfers the contents of S straight into A since OMsS is true. If d1 = 1, gQ is also opened so that R is transferred straight into Q. In the case of divide orders d1 = 0, so gQ is bypassed leaving Q unchanged and in fact zero. This K2 request also opens DgE and thus transfers the new accumulator exponent, e + 1, into E. If z = 0 and dov = 1 as discussed above, OV is set to "1". However, if z = 1 but dov = 1, OV is not set to "1" but is bypassed. This is to prevent setting OV to "1" when Z is already set to "1". (This rule is violated when the accumulator contains zero and a VID order is executed. The exit to G1 via D5 sets OV to "1".) If dov = 0, OV is always bypassed. At the same time, the Eccles-Jordan in G1 or K3 is set to "1" depending on whether TC is set to "0" or "1".

When $gAr \vee gQr \vee DgEr \vee sAQr \vee OVr \vee (G1\text{-}K2a)[(K2) \vee (\overline{tc}) \vee K3\text{-}K2] = 0$, the K2 Eccles Jordan is set to "0". After all the requests and replies have returned to "1", the active phase of G1 or K3 is initiated.

Now consider the case where K2 is active and c = 0. If the outputs of both the S and R zero detectors are "1", $\overline{SZ} \vee \overline{RZ} = 0$, so Z is set to "1". Otherwise, Z is bypassed. The S and R zero detectors appear on ZDR (D-1505). OV is always bypassed in this case. The Eccles-Jordan in G1 or K3 is set to "1" depending on whether TC is set to "0" or "1".

When $Zr \vee (G1\text{-}K2b)[(K2) \vee (tc) \vee (K3\text{-}K2)] = 0$, the K2 Eccles-Jordan is set to "0". As before, when all K2 requests and their corresponding replies return to "1", the active phase of either G1 or K3 is initiated.

Since the G1 control step is described in section 5.1, we consider K3. This control step is active only during the first pass through (K) in the execution of an SSC or ASC order. Its primary purpose is to open the OR gate to EXR and set the R1 Eccles-Jordan to "1" as shown on D-1282. EXR is set to "1" if $(\overline{z} \vee \overline{d1})(\overline{n\alpha})(\overline{za} \vee d1)(n \vee \theta_1 \vee \theta_2 \vee \overline{QZ})(\overline{j} \vee \overline{QZ}) = 0$. Since K3 may be activated from K1, CR may still be set to "1". As discussed in section 3.1.12, it is necessary to set CR to "0" to insure that α'_{-3} and α'_{-2} represent the true sign bit of A in the event of a right shift into S with KgA = 1. It so happens that this condition never arises in passing through (R) and (S) in completing and SSC or ASC order. It may happen during the second pass through (K) on the way to G1, but if so, CR is set to "0" during K1 or G1. As an afterthought, therefore, the request to set CR to "0" in K3 is not necessary. The $OgQ_{43}Q_{44}$ request is necessary to make SSC and ASC consistent with the order pairs SUB, STC and ADD, STC.

When the R1 Eccles-Jordan is set to "1", EXRr = 0, and CRr = 0, R1-K3 changes from "1" to "0" and sets the K3 Eccles-Jordan back to "0". When the K3 requests and replies return to "1", the R1 control step is activated. Its requests are discussed in section 5.10.

5.4 The Load Q Sequence (L)

This sequence is used only by the LAL order. Its function is to load Q (i.e., the least significant half of the accumulator) with a fractional operand but leave A unchanged. The accumulator exponent in E is also left unchanged. Since Z is set to "0" even though A may contain zero, control

passes through (K) on the way to G1. If A is zero and a zero fraction was placed in Q by (L), then K2 detects this and sets Z to "1". Thus, when G1 is activated to decode the next instruction, Z is set to "1" if AQ contains zero in any form. If the exponent in E is less than -128 at the beginning of an LAL order and the operand placed in Q is non-zero, then K2 will not set Z to "1" even though the entire accumulator is considered zero under other circumstances.

The description of the (L) sequence and each of the remaining sequences is abbreviated. The effect of each control step on the AU and LM is discussed, but a description of the signal flow through the sequence logic is omitted.

The G1 entry to L1 is shown on D-1280. Since KgA = 1 is an initial condition for G1 and since the state of the MsA mechanism is not changed during G1, α represents the assimilated contents of A at the time L1 becomes active.

During L1, α is placed in R and S is cleared to zero. MsS is requested so that the fractional operand in M appears as σ (i.e., output of the S-adder) by the time L2 becomes active. Z is set to "0" to cover the case where Z is currently set to "1" but the operand in M is non-zero.

As mentioned above, this does not cover the case where Z is true because of exponent underflow at the time of entry to L1.

During L2, 4σ is placed in A. This means the sign bits of the fractional operand in M are lost. Furthermore, r_{-1} and r_0 are placed in A_{43} and A_{44}. These bits, incidentally, are equal to the sign bit of α as it appeared at the output of the A-adder during L1.

In L3, the contents of R are transferred to M via RgM as discussed in section 3.1.2.2. During this step, the zero stored carry bits in A flush through the carry generator and A-adder logic. After this

occurs, the new contents of A (i.e., the fraction that is to be placed in Q) appear as α.

In L4, α is placed in R. Remember that $\alpha_{-1} \rightarrow R_{-1} \ldots \alpha_{44} \rightarrow R_{44}$. Note also that $\alpha_{43} = \alpha_{44}$ represents the true sign bit of the accumulator as it appeared during L1 and is therefore noise as far as the fractional operand from M is concerned. During L3 and L4, the new contents of M -- namely the old assimilated contents of A -- appear as σ since S contains zero and MsS is still true.

During L5, SRsAQ is requested as the new state for sAQ and both gA and gQ are opened. As a result, σ is placed in A and the contents of R are placed in Q. In this transfer $\sigma_i \rightarrow A_i$ and $r_i \rightarrow Q_i$.

At the completion of L5, the original -- possibly unassimilated -- contents of A are returned to A in assimilated form, and the nonsign bits of the fractional operand that was placed in M during G1 are in Q_{-1}, $Q_0 \ldots Q_{42}$. Q_{43} and Q_{44} hold the same bit as A_{-1} and A_0 at this point. However, Q_{43} and Q_{44} are cleared to zero by $0gQ_{43}Q_{44}$ during the G1 following (K). Z is set to "1" during K2 if AQ contains zero.

5.5 The Add Sequence (A)

The (A) sequence performs floating point addition (subtraction). The number in AQ which may be unassimilated during G1, and its exponent in E correspond to the augend (minuend) while the fraction in M and its exponent in EM correspond to the addend (subtrahend). Before the addition (subtraction) is performed, the augend (minuend) is shifted in AQ so that its exponent agrees with the exponent of the addend (subtrahend). In case the addend (subtrahend) is much smaller than the augend (minuend), the fraction in M is placed in A and shifted right until the exponents agree.

Let the exponent of the augend (minuend) be denoted by ea. Even though it is in E at the beginning of G1, it is transferred to EA during the decode step. Since the exponent of the addend (subtrahend) is placed in EM during decode, let it be denoted by em. $\overline{EM}$sD is the state of the sD mechanism and C is set to "0" at the time (A) is entered from either G1 or V4. Therefore, during A1 the output of the exponent adder is d = ea - em - 1. The five cases of floating addition (subtraction) are defined in terms of d at this point. Note that d is always a unit less than the true difference between the exponents of the augend (minuend) and addend (subtrahend).

Case 1: fa #1 = $(d \leq -46) \vee dz = 1$ (i.e., condition is true)

Case 2: fa #2 = $(-45 \leq d \leq -1) = 1$

Case 3: fa #3 = $(0 \leq d \leq 21) = 1$

Case 4: fa #4 = $(22 \leq d \leq 43) = 1$

Case 5: fa #5 = $(44 \leq d) \vee dov = 1$

It is convenient to decode cases 2 and 3 together.

Case 2, 3: fa #2, 3 = $(-45 \leq d \leq 21) = 1$

The signals fa #1, fa #2, 3, fa #4, fa #5, dov and dz are outputs of the exponent decoder as shown on D-1504 and discussed in section 3.2.4. Remember that if dz = 1, then $d < -128$. If dov = 1, then $d > 127$.

If fa #1 = 1, the actual difference in exponents, $d' = d + 1$, is $d' \leq -45$. This means the augend (minuend) in AQ must be shifted right at least 45 base 4 places before addition (subtraction) occurs. Since there are only 45 nonsign base 4 positions in AQ, the fraction in M is simply clear added (subtracted) via the (B) sequence. Consequently, if the original accumulator was negative, the sum (difference) is too large by at most 2^{-90}.

If fa #2, 3 = 1 and $d < 0$, the actual exponent difference is in the range: $-44 \leq d' \leq 0$. Case 2 requires that the contents of AQ be right

shifted from 0 to 44 base 4 positions, before addition (subtraction) occurs. The sum (difference) has the exponent of the addend (subtrahend), i.e., the exponent in EM.

If fa #2, 3 = 1 and $d \geq 0$, the actual exponent difference is in the range: $1 \leq d' \leq 22$. Case 3 therefore requires left shifting the contents of AQ 1 to 22 base 4 positions before addition (subtraction) occurs. Since the sum (difference) has the exponent of the augend (minuend), i.e., the exponent in E, the partial result must be right shifted the same number of base 4 positions with the appropriate propagation of carries or borrows. Circular left and right shift paths at the high end of A and S and the low end of Q and R are used to accomplish this. As discussed in sections 3.1.2.6, 3.1.2.8, and 3.1.12, these paths are opened by setting CL and CR to "1".

If $d' = 22$, the contents of AQ are circularly left shifted until the two least significant nonzero bits that were originally in Q_{41} and Q_{43} are placed in A_{43} and A_{44}. The α_{-1} and α_0 bits that appeared at the output of the A-adder during G1 with KgA = 1 are in Q_1 and Q_2 at the time of addition (subtraction). As the result is circularly right shifted, the Q and R half-subtractors (section 3.1.12) propagate a borrow to the left if necessary. The rate at which a carry is propagated to the left is exactly equal to the rate at which the result is shifted right. Thus , as long as a carry is being propagated to the more significant bits of the result, A_0^* and S_0^* both contain a "1".

If fa #4 = 1, the actual exponent difference is in the range: $23 \leq d' \leq 44$. Consequently, the contents of AQ would have to be left shifted from 23 to 44 base 4 positions. This is generally not possible using the circular shift techniques described above. Instead, the contents of A and Q are interchanged, and A is isolated. The new contents of A and the contents of M are then interchanged with -M being placed in A if a subtract

order is being executed. This is done by steps A2 through A7 in (A). The addend (subtrahend) that is now in A is right shifted $d' - 23 = d - 22$ base 4 positions -- leaving Q unchanged -- before addition (subtraction) occurs. The result represents the least significant half of the sum (difference) with a possible carry or borrow to be propagated into the most significant half of the addend (minuend) which is presently in Q. After A and Q are transferred to R and S respectively, the carry-borrow logic described in section 3.1.13 determines whether $+2^{-44}$ or -2^{-44} should be added to the contents of S (i.e., the most significant half of the addend (minuend)). The final sum (difference) is gated straight into AQ and assigned the exponent of the augend (minuend). This final interchange and carry-borrow correction occurs in steps A11 through A14 of the (A) sequence as shown on D-1269.

If fa #5 = 1, the actual exponent difference is: $d' \geq 45$. This means the quantity in M should be added (subtracted) to those bits of AQ lying to the right of Q_{42}. If $d' = 45$, the bits in M_{-1} and M_0 should be added (subtracted) to the bits in Q_{43} and Q_{44}. Since these bits are always set to "0" during G1, the (A) sequence is simply bypassed in this case and the augend (minuend) with its exponent is accepted as the sum (difference) even though Z may be set to "1".

The descriptions given above apply to SUB, ADD, SSC, and ASC. It should be clear that if the addend (subtrahend) in M is zero (i.e., em = -64) the (A) sequence is bypassed and the augend (minuend) is always taken as the sum (difference). In the step-by-step description given below, note that when d is odd (d' even) the addition (subtraction) is to A while if d is even, it is to S.

In A1, the negative conditional logic shown below is used to determine the particular case of addition. This logic and the first eight

control steps of A appear on D-1268.

$$\text{FA } \#1 = (\text{em} = -64) \vee \overline{z}(\overline{\text{fa } \#1}) = \text{A1-B1}$$

$$\text{FA } \#2,\ 3 = (\text{em} = -64) \vee z \vee (\overline{\text{fa } \#2,3}) = \text{A1-A8}$$

$$\text{FA } \#4 = (\text{em} = -64) \vee z \vee (\overline{\text{fa } \#4})$$

$$\text{FA } \#5 = (\overline{\text{em} = -64})[z \vee \text{fa}\#5] = \text{A3-K1}$$

If FA #1 = 0, the Eccles-Jordan in B1 is set to "1" since a CSB or CAD is required either because Z is set to "1" or fa #1 = 1. After the A1 Eccles-Jordan is set to "0" by B1-A1, the B1 control step becomes active. Note that nothing is done to the LM or AU during this option of the A1 step. It is designated as a "null" step on the DC Flow Chart. This step is necessary because B1 clears EA to zero, which may cause the fa #1 output of the decoder to change from "1" to "0" and thus destroy the entry request. In other words, if the B1 requests were made in series with entry request, FA #1 = 0, without setting the B1 Eccles-Jordan, FA #1 might change from "0" to "1" before the B1 step is completed and thus cause an error. By setting the B1 Eccles-Jordan to "1" first, FA #1 may change from "0" to "1" during the active phase of B1 and not affect the result. The operation of the (B) sequence is discussed in section 5.2.

If FA #2,3 = 0, neither the accumulator nor the operand are zero, but fa #2,3 = 1. The A8 Eccles-Jordan is not affected, but the A9 Eccles-Jordan is set to "1" and the MsA mechanism is set according to the output of the θ decoder (see section 3.1.14) if d = -1. In this case, addition (subtraction) occurs immediately because the exponents of the accumulator and operand agree. The A9-m signal sets the A1 Eccles-Jordan back to "0" in this case. The activity in the add loop (A9 and A10) is described after the last two options of A1 are considered.

If FA #4 = 0, neither the accumulator nor the operand are zero, but fa #4 = 1. As indicated above, a preliminary shuffle of the contents of A, Q, and M is necessary in this case. The details of this shuffle are given below.

In A2 -- which is a particular extension of the A1 request -- the assimilated contents of A (KgA = 1) are placed in R while the contents of Q are placed in S. OMsS is set so the new contents of S appear as σ. In the EAU, d = ea - em - 1 is placed in ES. The A4 Eccles-Jordan is set to "1". When all replies are "0", the A1 Eccles-Jordan is set to "0", causing the A2 request to go to "1".

When A4 becomes active, the contents of S and R are transferred straight into A and Q. OMsA is set so that the new contents of A appear as α. In the EAU, d is transferred from ES to EA and 22 is subtracted from it by setting -22sD. Consequently, the new output of the exponent adder is $d_1 = d - 22$ where d = ea - em - 1 during A1.

In A5, α is placed in R and zero is placed in S. The MsS mechanism is set according to the output of the θ decoder so that $\sigma = -m$ if SUB or SSC is being executed, while $\sigma = m$ is ADD or ASC is being executed. At the same time, d_1 is placed in ES.

During A6, $\pm$ m is placed in A and θ_1 is set to "1", indicating that the new contents of M are to be added to the contents of A. DL is set to "0" to insure that gQ and gR are bypassed during A9 and A10. This isolates the most significant half of the augend (minuend) which is held in Q. In the EAU, zero is placed in EA and d_1 is placed in EM. $\overline{\text{EM}}$sD is set with C still set to "0" so that $-d_1 - 1 = -ea + em + 22$ appears at the output of the exponent adder. Since $-22 \leq -d_1 - 1 \leq -1$, the quantity in A (i.e., $\pm$ m) is right shifted by the add loop (A9, A10) as in a case 2 addition.

A comment is in order concerning $-d_1 - 1$. Since 23 left shifts of the augend (minuend) were effectively performed by the interchange of A and Q, the remaining number of left shifts for Q is given by $0 \leq ea - em - 23 \leq 21$. Instead, the addend (subtrahend) is right shifted by the same amount. Because the conditional logic of A9 and A10 is based on an exponent difference that is one less than the true difference, $-(ea - em - 23) - 1 = -ea + em + 22$ must appear at the output of the D-adder at the time A9 is entered.

In A7 the least significant half of the augend (minuend) is placed in M. Control passes to A8 where the activity was described in connection with the FA #2,3 option of A1.

If FA #5 = 0 in A1, the request A3-K1 bypasses (A) and sets the K1 Eccles-Jordan to "1". In this case, the operand is zero or the accumulator is not zero and fa #5 = 1. A null step between A1 and K1 (i.e., A3) is necessary because the gEA in K1 may remove the entry request. The K1 - b signal sets the A1 Eccles-Jordan to "0" which then activates the K1 control step.

We now consider the add loop which contains the A9, A10, and A11 control steps as shown on D-1269. When the A9 Eccles-Jordan is set to "1", the output of the D-adder lies in the range: $-45 \leq d \leq 21$. If $-45 \leq d \leq -1$, AQ or A is shifted right d + 1 base 4 places prior to the addition (subtraction). Bits are lost at the right end of Q or A depending on whether DL is set to "1" or "0". If $0 \leq d \leq 21$, AQ is circularly left shifted by d + 1 base 4 places prior to the addition (subtraction). The result is then circularly right shifted with carries or borrows propagated to the left.

Since A9, A10, and A11 constitute an inner control loop, it is necessary to consider the signal flow through it. The entry logic is

somewhat nonstandard in order to start shifting AQ as quickly as possible when FA #2,3 = 0. For example, if d ≠ -1 in A8, the A9 Eccles-Jordan is set to "1" immediately. Its "0" output, A9- m is used to set the A1 Eccles-Jordan to "0" and thus cause A8-A9 to change from "0" to "1". However, the output of the A9 $\overline{A}^*$ goes to "0" as soon as the Eccles-Jordan in A9 is set to "1" and does not wait for A8-A9 to change from "0" to "1". In all other loops the signal which is analogous to A8-A9 also feeds the $\overline{A}^*$ so that it must change back to "1" before the first step in the loop can become active. There are two cases in which the start of A9 is inhibited in this manner. If FA #2,3 = 0 and d = -1, the MsAr must to to "0" before the A9 Eccles-Jordan is set to "1", so the A9 $\overline{A}^*$ cannot go to "0" until the A1 Eccles-Jordan has been set to "0" and MsAr returns to "1". A similar statement applies if the A8 Eccles-Jordan is set to "1" at the time A8-A9 changes from "1" to "0".

In all control loops the Eccles-Jordan that is analogous to A9 is set to "1" when the loop is entered. It remains at "1" as long as the exit condition is not satisfied (i.e., ex = 0). The ex and $\overline{ex}$ signals feed the $\overline{A}^*$s analogous to A11 and A9 respectively. These signals may come from the EXA, EXF, or EXR status memory elements as discussed in section 4.3. As long as ex = 0, the output of the A11 $\overline{A}^*$ or its equivalent is held to "1" while the A9 and A10 $\overline{A}^*$s or their equivalent alternately become active and relax. When ex changes to "1" during A10, the output of the A9 $\overline{A}^*$ is held to "1" and the output of the A11 $\overline{A}^*$ goes from "1" to "0" instead. This ultimately sets the A9 Eccles-Jordan back to "0".

While control alternates between A9 and A10, the A10 Eccles-Jordan is alternately set to "0" or "1". When the A9 $\overline{A}^*$ first becomes active, the A10 Eccles-Jordan is set to "0". The replies from the link mechanisms

requested by A9 go to "0" and set the A10 Eccles-Jordan to "1". This cannot occur unless the A9 Eccles-Jordan is set to "1" (i.e., not unless A9 - m = 0). When these replies return to "1", the output of the A10 $\overline{A^*}$ goes to "0". The link mechanisms which it requests send back "0" replies which cause the A10 Eccles-Jordan to set to "0". The A10 request and associated replies return to "1". If exa = 0, A9 becomes active and the process is repeated. If exa = 1, A11 becomes active and sets the Eccles-Jordan in either K1 or A12. This combined with "0" replies from CL and possibly MsS set the A9 Eccles-Jordan back to "0". When the add loop is terminated, the A9 and A10 Eccles-Jordans are both set to "0".

The conditional logic associated with A9 and A10 is complicated by the presence of bypass signals called "bya" and "byb". These signals are used to generate bypass requests for selector mechanisms and certain status memory elements after the shift paths have been established and the only activity is opening and closing the gates in the MAU to shift and in the EAU to count. The negative logic for these signals is given below.

$$\overline{\text{bya}} = (\overline{\text{ESsEA}}) \vee \overline{\text{edc}}$$

$$\overline{\text{byb}} = \Omega \vee (\text{es} = -3) \vee (\text{es} = -2) \vee (\text{es} = -1) \vee (\text{es} = 0)$$
$$\vee (\text{es} = 1) \vee (\text{es} = 2)(\text{cr})$$

The ESsEA signal is generated by the sEA mechanism as shown on D-1185. The Ω and cr signals are outputs of the status memory elements of the same name. The edc signal and the (es =) signals are outputs from the EDC F-element and ED register as shown on EDM (D-1504).

Perhaps the best way to describe the action during A9 and A10 is to consider two examples. In the first example let d = -3 so that AQ or A is right shifted 2 places before addition (subtraction) occurs. In the

second example let d = 2 so that AQ is circularly left shifted 3 places before addition (subtraction) occurs and then circularly right shifted the same number of places.

Assume d = -3 when A9 first becomes active. Note bya = 0 since the sEA selector is set to EsEA during G1. Remember KgA was true during decode and is still true. The sSR mechanism is set to 1/4 AQsSR and CL is bypassed. The MsS mechanism is set to OMsS. If case 2 addition applies, DL is set to "1" so both gS and gR are opened causing the assimilated contents of AQ to be right shifted into SR with the sign bit, α_{-1}, duplicated in S_{-1} and S_0. If case 4 addition applies (case 2 or case 4 must apply because $d < 0$), DL is set to "0" so only the assimilated contents of A (i.e., α) is right shifted into S with α_{43} and α_{44} being lost. In either case, the new contents of S appear as σ at the output of the S-adder because OMsS is set.

In the EAU, gES is opened so that -3 is placed in ES. Note that gES always gates the ED register as well. Since ESsEA is not true during the first pass through A9, Ω is set to "1". If d1 = 1, EMsE is set and gE opened so the exponent of the addend (subtrahend) is transferred to E as the exponent of the sum (difference). If d1 = 0, case 4 applies and the exponent of the augend (minuend) in E is also the exponent of the sum (difference), so E is left unchanged -- gE and sE are bypassed. With this accomplished, control passes to A10.

During this first pass through A10, byb = 0 because es = -3. Both gA and gQ are opened if d1 = 1, only gA is opened if d1 = 0. The sAQ mechanism is set to 1/4 SRsAQ and the MsA mechanism is set according to the θ decoder output. Addition (subtraction) occurs at this point because the accumulator has been shifted right 2 places as required. Therefore, α represents the unassimilated sum (difference). It is necessary to pass through A9 and A10 again to place this sum (difference) in A.

In the EAU, gEA is opened at the same time as sEA is being set to ESsEA. Thus, -3 is placed in EA. Note that the sEA mechanism was previously set to EsEA or OsEA. Therefore, depending on the timing of the gEA signal, the initial inputs to the EA register may be incorrect. The gEA signal also opens the EDC F-element gate. In this case the EDC F-element is set to $EDC = (\overline{es = -4})(\overline{es = 0})(\overline{es = 1})(\overline{es = 2}) = 1$. The ESgEM gate is bypassed because es = -3. CR is bypassed and 2sD is selected. This causes the output of the exponent adder to become -1 since C was set to "0" during decode.

The sum (difference) is not in AQ or A at this point, so the "done" signal is "0". In positive logic:

$$done = (es = -2) \vee (es = -1) \vee \Omega(es = 0) \vee cr(es = 2)$$

When "done" is "0", EXA is again set to "0". Notice that EXA is set to "0" during G1 and V4 as well to prevent immediate exit from the add loop. With exa = 0, the A9 step again becomes active.

During this second pass through A9, bya = 1 because sEA is set to ESsEA and edc = 1. Therefore, gE, sE, MsS, sSR, and CL are automatically bypassed with selector mechanisms and memory elements retaining their old state. Since bypassing yields fast replies, this pass through A9 is termed "fast". If both the gS and gR gates are opened, the unassimilated α representation and the contents of Q are shifted right into SR with q_{43} and q_{44} being lost. If only gS is opened (i.e., dl = 0), α_{43} and α_{44} are lost. This is of no consequence since these bits are cleared to "0" during the following decode step. The gES gate is opened, so -1 is placed in ES. $(\overline{EMsD})(ESsEA) = 1$ in positive logic, so Ω is set to "0". With this accomplished, control passes to A10.

During this second pass through A10, es = -1 so byb = 0. The

sAQ mechanism is set to 4SRsAQ. If gA and gQ are both opened, the sum (difference) is left shifted into AQ with "0" being placed in Q_{43} and Q_{44}. If dl = 0, only gA is opened and the single length sum (difference) is left shifted into A with "0" being placed in A_{43} and A_{44}. The "done" signal is "1" because es = -1, so the MsA mechanism is set to KgA, and EXA is set to "1".

In the EAU, gEA places -1 in EA. The sD mechanism is again set to 2sD, so d = +1 although it is not used. ESgEM is again bypassed. With exa = 1, A11 now becomes active instead of A9.

If dl = 1, case 2 addition was performed so CL is set to "0". This action is unnecessary in this instance but necessary for case 3 addition. Control now passes to K1. The (K) sequence may find α overflowed (i.e., αov = 1) or AQ zero.

Assume dl = 0 at A11. This implies that case 4 addition was performed, so some terminal operations are necessary. CL is set to "0" only to obtain a "0" reply, since CL is never set to "1" in A9 when dl = 0. The MsS mechanism is set to OMsS for use in A12. The carry-borrow logic meanwhile has examined the α_{-1}, m_{-1}, and m_0 bits as described in section 3.1.13. The gCB gate is opened in A11 to allow the output of this logic to be stored in the CB F-element.

In A12 the most significant half of the augend (minuend) is placed in S and the least significant half of the sum (difference) is placed in R. A carry, a borrow, or nothing is now added to the $44^{\underline{th}}$ position of S. The positive conditional logic which determines whether a carry, a borrow, or nothing is added appears below.

$$(cb)(\bar{m}_0) = 1 \quad \rightarrow \quad \text{Add } 2^{-44}$$

$$(cb)(m_0) = 1 \quad \rightarrow \quad \text{Add } -2^{-44}$$

$\overline{cb} = 1 \rightarrow$ Add Nothing

The cb signal is the true output of the CB F-element and m_0 is the $0^{\underline{th}}$ bit of the M register. Since the M register contains the least significant half of the augend (minuend), m_{-1} and m_0 both have positive weight at this point and are not necessarily equal. To add 2^{-44}, By $M\overline{M}$ and CS requests are sent to the MsS mechanism. (See section 4.2.) To add -2^{-44}, $M\overline{M}sS$, $\overline{CS}$ is the setting requested. If nothing is added, By $M\overline{M}$ and By CS requests are made to cause $M\overline{M}r = CSr = 0$, but the state of the MsS mechanism is left at OMsS as established in A11. The most significant half of the sum (difference) appears as σ.

During A13, σ and the contents of R are transferred straight in AQ, and DL is set to "1". Since KgA = 1, the assimilated representation of A appears as α.

In A14, the MsS mechanism is cleared to the OMsS state by a KMsS request. This is necessary if MsS was set to $M\overline{M}sS$ or CS during A12 as discussed in section 4.2. Control then passes to K1.

We now return to the second example. Assume d = 2 when A9 becomes active for the first time. This implies a case 3 addition in which AQ is circularly left shifted by three places, M is added (subtracted), and the sum (difference) is circularly right shifted three places. This means the addition (subtraction) is done with the S-adder. Furthermore, in this particular case bya = byb = 0 for all steps in the loop.

During the first pass through A9, 4AQsSR is set and CL is set to "1". As discussed in section 3.1.2.12, this opens the paths $\alpha_{-1} \rightarrow R_{43}$ and $\alpha_0 \rightarrow R_{44}$. The gS and gR gates are both opened, so a circular left shift of one base 4 position occurs. Note that KgA = 1 so α is assimilated. Ω is set to "1". The ES register is set to +2 when gES is opened. The gE and EMsE requests are bypassed because the exponent of the augend (minuend) in E is also the exponent

of the sum (difference). The MsS mechanism is set to OMsS. Control passes to A10 when this is accomplished.

During the first pass through A10, $_4$SRsAQ is set since CR was set to "0" during decode and es = 2. Because CL is set to "1", the paths $\sigma_{-1} \rightarrow Q_{43}$ and $\sigma_0 \rightarrow Q_{44}$ are now open. When gA and gQ are opened, another circular left shift is performed, placing the result in AQ. The four most significant bits of the assimilated augend (minuend) are in Q_{41}, Q_{42}, Q_{43}, and Q_{44} at this point. OMsA is set and EXA is set to "0" because es = 2 and CR is set to "0".

In the EAU, +2 is placed in EA and EM from ES. The latter transfer is made to save the count for the number of circular right shifts necessary following the addition (subtraction). The sD selector is set to 2sD while CR is bypassed -- meaning that CR retains its "0" setting. As a consequence of -2sD and C being set to "0" during decode, d = 0 now appears at the output of the exponent adder. Control returns to A9 since exa = 0.

During this second A9 step, another circular left shift is performed with the MsS mechanism being set according to the output of the θ decoder. The sum (difference) then appears as σ in unassimilated form. The gES gate places d = 0 into ES. Ω is set to "0". At the same time, the R half-subtractor logic begins to inspect the outputs of R_{43}, R_{44}, S_{-1}, and the t_{-1} input to the high S special adder. As indicated in section 3.1.12, this information is sufficient to determine whether a unit must be borrowed from the bit of the augend (minuend) that is now in R_{44}. If so, a unit is borrowed from R_{43}, R_{44} during the next A10 step in which a circular right shift of one base 4 position occurs. If both R_{43} and R_{44} contain "0", the borrow is propagated by placing a bit in A^*_{44} during A10.

During the second A10 step, 1/$_4$SRsAQ is requested and CR is set to "1". OMsA is requested and EXA is set to "0" because (es = 0) and Ω = 0. Note that even though cr changes from "0" to "1", 1/$_4$SRsAQ is set because es = 0. Since

cr = 1, the paths $\sigma'_{-3} \rightarrow A_{-1}$ and $\sigma'_{-2} \rightarrow A_0$ are now open. Therefore, when gA and gQ are opened, the sum (difference) is circularly right shifted one base 4 position into AQ with borrow propagation. The signals σ'_{-3} and σ'_{-2} represents the outputs of the R half-subtractor as discussed in section 3.1.12. If a borrow is necessary, $r_{43} = r_{44} = 0$, $\sigma'_{-3} = \sigma'_{-2} = 1$ and Q^*_{44} is set to "1". Otherwise, Q^*_{44} is set to "0".

In the EAU, gEA is performed while ESgEM is bypassed. EMsD is requested, so d = +2 since EA now contains 0 and C was set to "0" in decode. Control returns to A9 because exa = 0.

During the third pass through A9, the sum (difference) is circularly right shifted one more base 4 position into SR with a unit borrowed from Q_{43}, Q_{44} if Q^*_{44} contains a "1". Since 1/4AQsSR is requested and CR is set to "1", the paths $\alpha'_{-3} \rightarrow S_{-1}$ and $\alpha'_{-2} \rightarrow S_0$ are now open. As discussed in section 3.1.12, α'_{-3} and α'_{-2} are two outputs of the Q half-subtractor. OMsS is requested since d = 2. Furthermore, Ω is reset to "1" because EMsD is the present state of the sD mechanism. The d = +2 output is placed in ES via gES. The CL memory element is bypassed but remains set to "1". Control now passes to A10 for the final step in the loop.

During the third and final pass through A10, the last circular right shift places the unassimilated sum (difference) in AQ. The "done" signal is now "1" because cr(es = 2) = 1. Thus, KgA is requested and EXA is set to "1". In the EAU, +2 is placed in EA, -2sD is requested. The D-adder output is now zero. Since Ω is set to "1" and es = 2, ESgEM is performed again, but this does not affect the count. Since exa = 1, control now passes to A11.

In A11 d1 = 0, so CL is set to "0" and control passes to K1. Since KgA was requested during the last pass through A10, α is assimilated, so the αov and αnz signals are reliable when K1 becomes active.

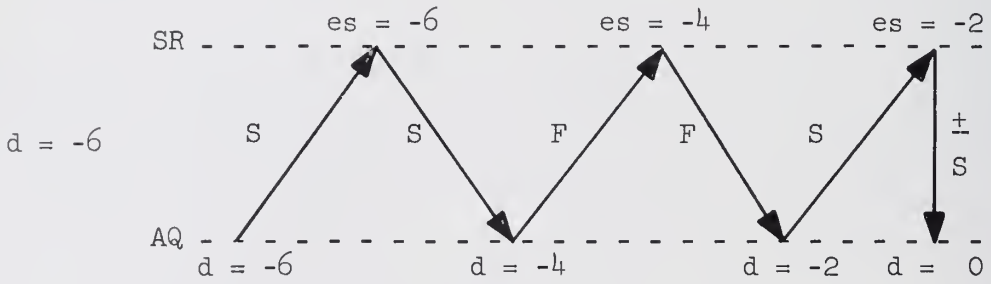
A short summary of the shift patterns in the add loop for $-6 \leq d \leq 5$ is given below. The letters "F" and S" denote "fast" and "slow" steps depending on whether the bypass signals bya and byb are "1" or "0".

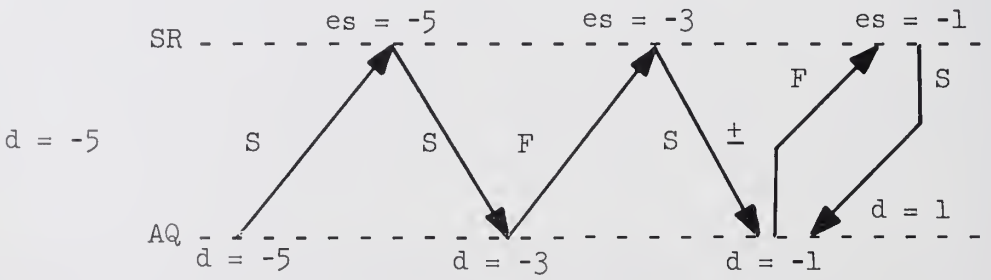
d = ea - em - 1

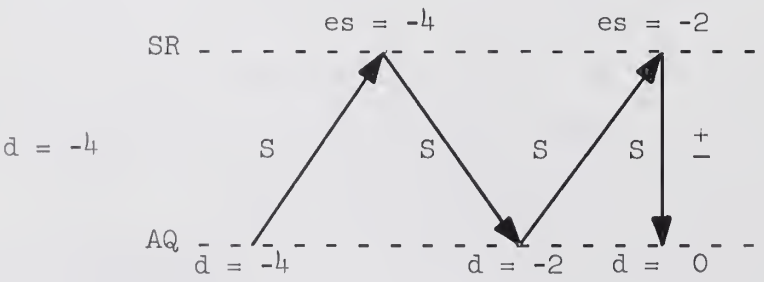
ea = augend (minuend) exponent

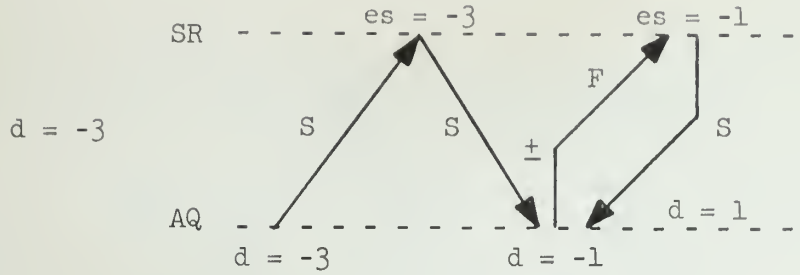
em = addend (subtrahend) exponent

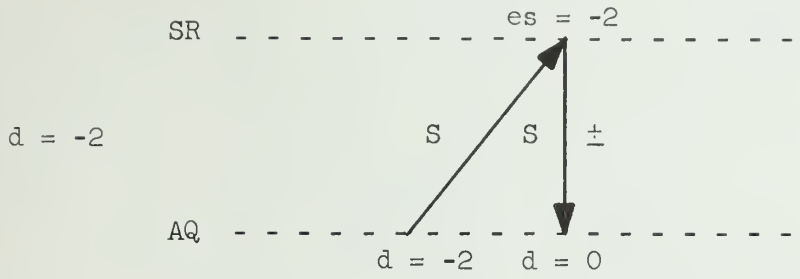
For Cases #2 and #4

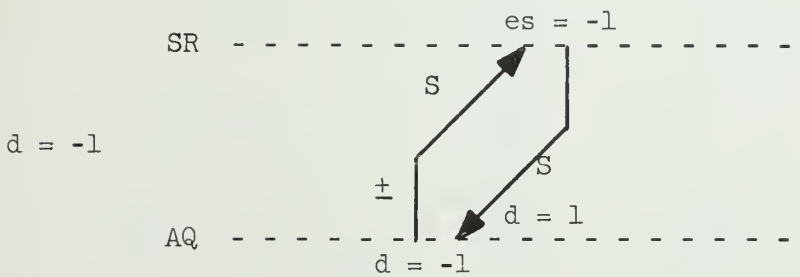


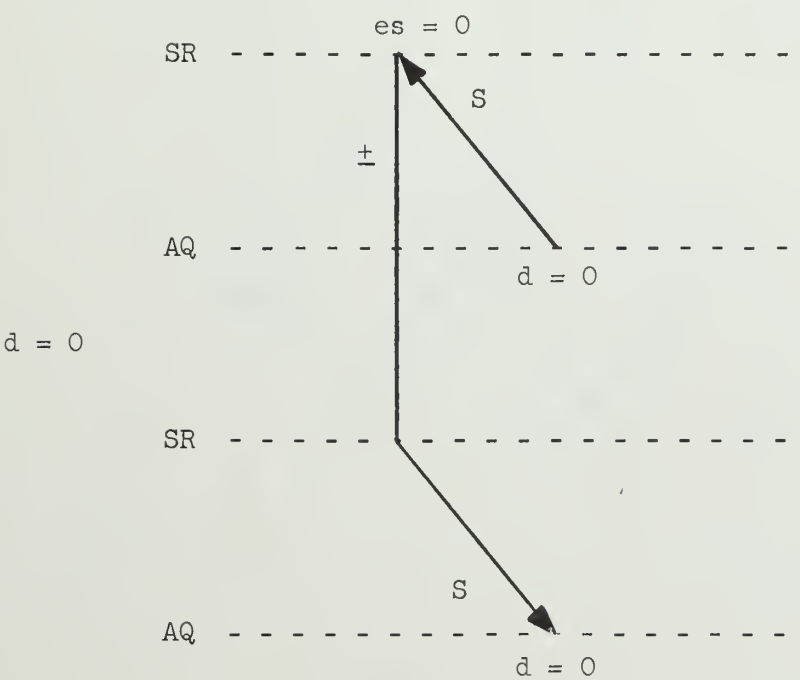


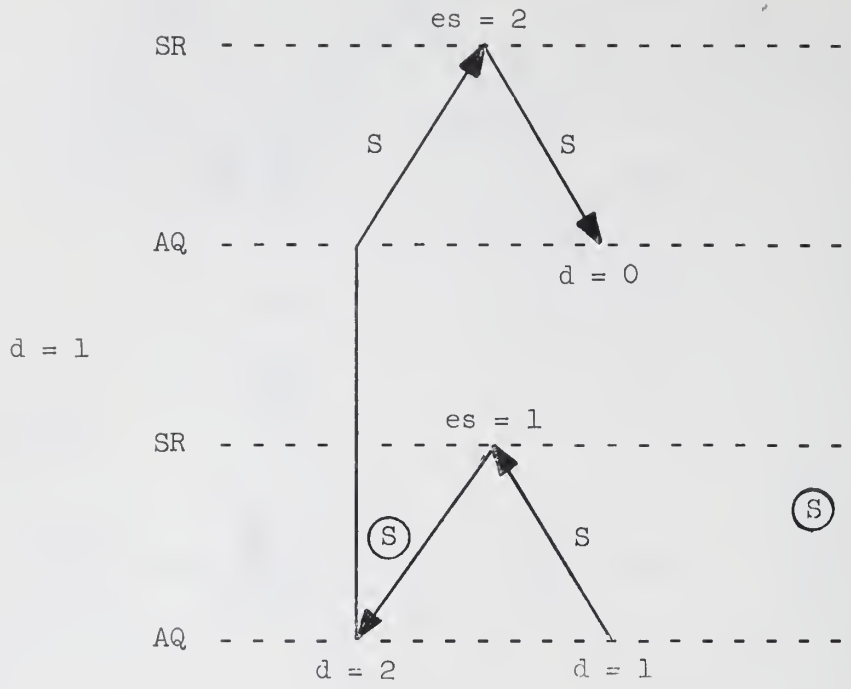


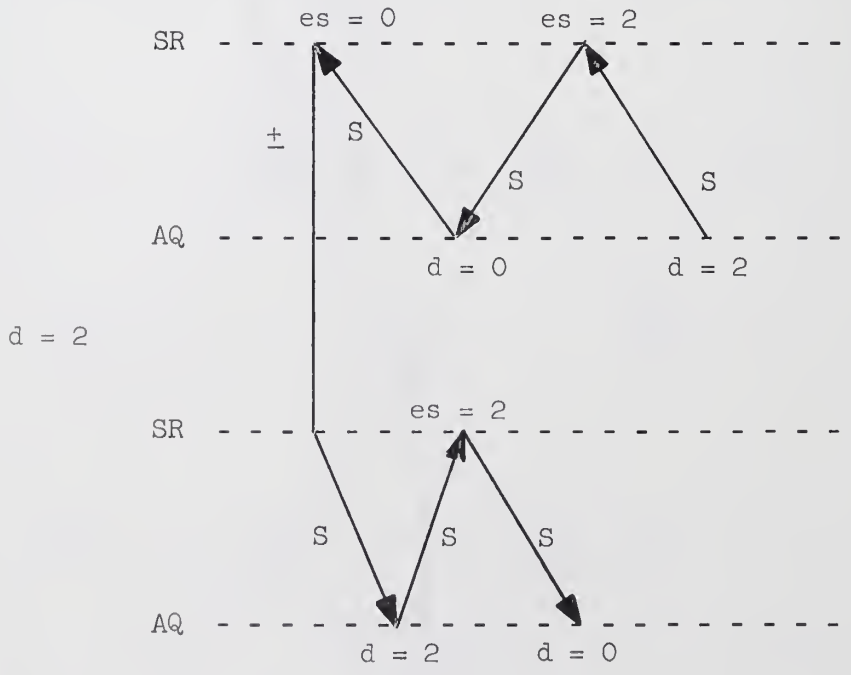

d = -3
SR
es = -3
es = -1
S
S
F
±
S
AQ
d = 1
d = -3
d = -1


d = -2
SR
es = -2
S
S
±
AQ
d = -2
d = 0

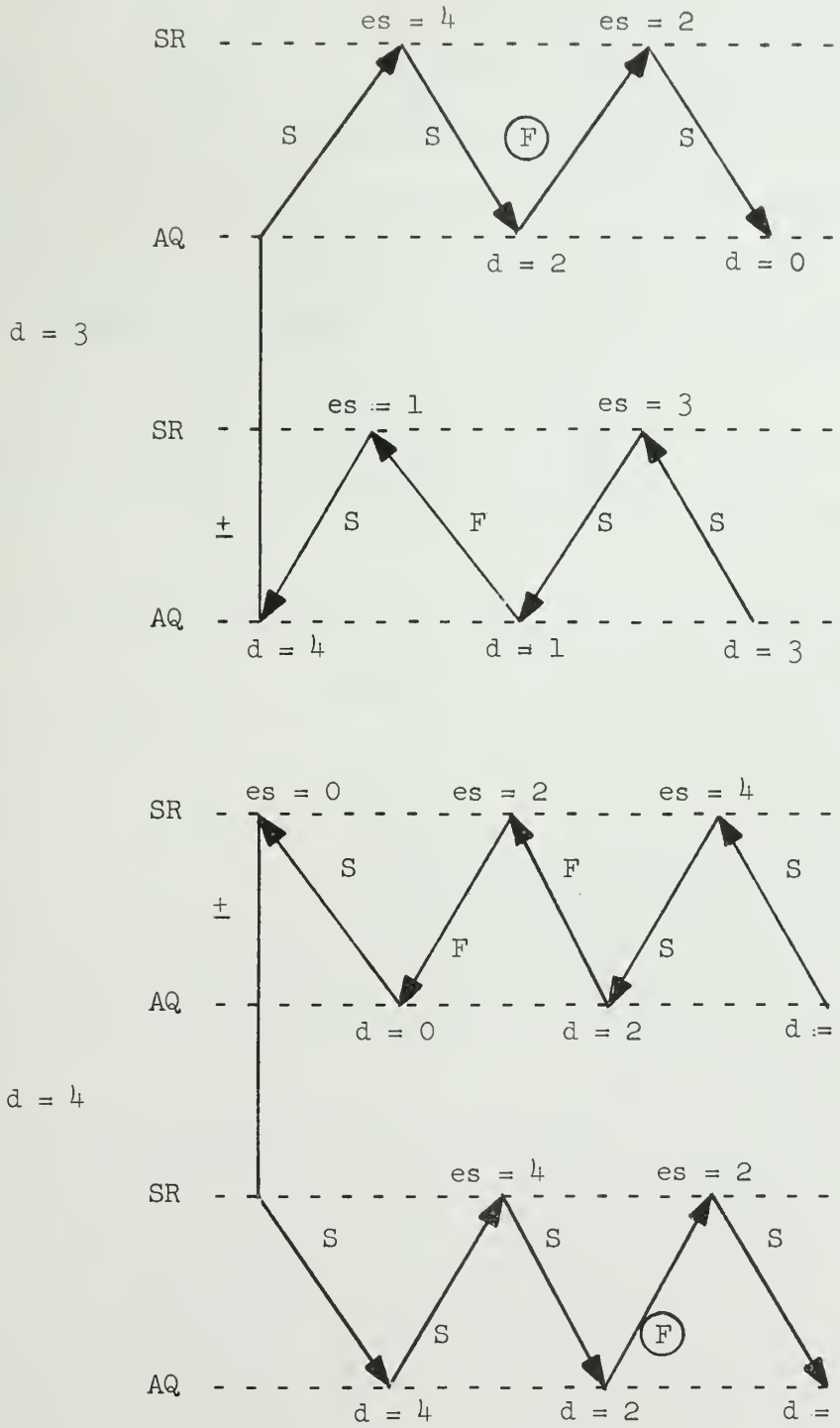

d = -1
SR
es = -1
S
S
±
d = 1
AQ
d = -1

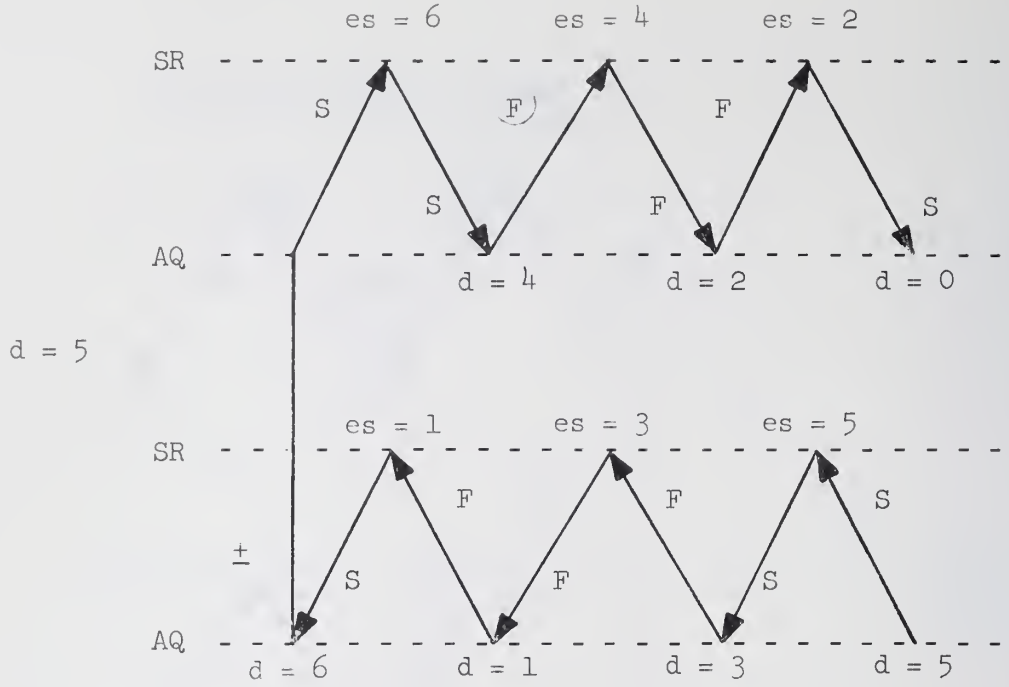

d = 0
es = 0
SR
S
±
AQ
d = 0
SR
S
AQ
d = 0


es = 2
SR
S
S
AQ
d = 0
d = 1
es = 1
SR
S
S
AQ
d = 2
d = 1
S
This shift is slow because ESgEM occurs with EMsD.


es = 0
es = 2
SR
±
S
S
S
AQ
d = 0
d = 2
d = 2
es = 2
SR
S
S
S
AQ
d = 2
d = 0

(F) This step is not truly fast since Ω is set to "0".





This summary shows that for $d \leq -7$ and $d \geq 6$, at least half of the steps in the add loop are faster as a consequence of the bypass signals, bya and byb.

5.6 The Exponent Arithmetic Sequence (E)

This sequence is used only by the SBE, ADE, CSE and CAE orders. It is entered directly from G1 in all cases as shown on D-1270. The exit from E3 always returns control to G1.

During G1, the C memory element and the sEA and sD selector mechanisms are set such that the D-adder output is:

d = ea - em	SBE
d = ea + em	ADE
d = -em	CSE
d = em	CAE

In this notation, ea represents the accumulator exponent which is also held in E.

The d output as defined above may change when E1 becomes active. In G1, the FlgMEM gate places Fl_{45} in both EM_6 and EM_7. At the same time, Fl_{44} is placed in M_{44}. If eight-bit exponents are being used, the bit in M_{44} is the true sign bit of the exponent in EM. To effect its transfer to EM_7, the $m_{44}gEM_7$ gate is opened when E1 becomes active. This may cause d to change.

When E2 becomes active, d is placed in E as the new exponent of the accumulator. This output is also inspected by the exponent decoder. If $d < -128$, a "1" is gated into the ESZ F-element of the ED register. This gating is done by gED = gES $\vee$ DgE as discussed in section 3.2.4.

When E3 becomes active, Z is set to "1" if esz = 1 and is bypassed otherwise. OV is set to "1" if esov = 1 and Z is set to "0". It is bypassed otherwise. Control then passes to G1.

5.7 The Store Sequence (S)

As shown on D-1283 and D-1284, the (S) sequence has entries from G1, P1, F^4, and R3. The normal entry to (S) is via S1. SAM and STU enter S1 directly from decode. STR, XCH, STC, and STN go through (G) and (R) and then enter S1. This is done to normalize the fraction in AQ (i.e., insure that $-1 \leq f < -1/4$, $f = 0$, or $1/4 \leq f < 1$) before A is rounded, assimilated and stored. Note that XCH involves a terminal clear add operation. The SSC and ASC orders are equivalent to SUB or CSB, STC and ADD or CAD, STC respectively except G1 does not become active between the subtraction or addition and the store clear operation. For STF, SIF, SEQ and SIA control enters the (P) sequence from G1 and then passes through (F) to S1. In (F) AQ is shifted right or left until its exponent is equal to the value specified (i.e., until e = 0 for STF, e = 22 for SIF, e = m during G1 for SEQ, and e = 6 for SIA). If e = 0 during G1 when STF is decoded, control passes from P1 to S1 directly. The first nonstandard entry is

S2. This entry is used by the SAL and SEX orders. The second nonstandard entry is S3 which is used only by the SRM order.

Consider AQ and E at the time S1 becomes active. Regardless of what preceded the S1 step, the assimilated contents of A and the seven least significant bits of E will be stored in FO (OUT) during S9 with or without modification due to fractional overflow, normalization, or zero corrections. The STU, STR, XCH, STC, STN, SSC, ASC, STF and SEQ orders round the contents of A prior to storage in FO. The round-off logic is discussed in section 3.1.7. It should be noted that round-off by itself does not change the S1 contents of A. However, the XCH, STC, STN, SSC, and ASC orders do change the S1 contents of AQ and sometimes the S1 contents of E for other reasons. Of the orders which enter S1 only SAM, SIF, and SIA do not round the contents of A prior to storage in FO. The fractional overflow, normalization and zero corrections are discussed below in conjunction with the ST step.

In S1 the RO status memory element is set to "1" for those orders which round A prior to storage in FO, i.e., those orders for which $\overline{\lambda}(\overline{j} \vee \overline{\theta}_2 \vee x) = 1$. As discussed in section 3.1.7, ro = 1 allows the round-off, ρ, to assume a value of "1" or "0" as determined by the bits in A_{44}, A^*_{44}, and Q. The RO memory element is set to "0" for the SAM, SIF, and SIA orders, i.e., the orders for which $\lambda \text{ " } j\theta_2\overline{x} = 1$. Note that the x variable is unnecessary in these expressions. To allow adequate time for the round-off, ρ, to change the output of the carry generator and A-adder, KgA is requested although the MsA mechanism is already set to the KgA state. This request is always made even though RO is set to "0". As pointed out in section 4.2, the MsAr response to this request is guaranteed to be delayed enough to insure the α-logic signals (such as αov and nα) are reliable when the next control step -- S4 in this case -- becomes active.

When S1 becomes active during the execution of an SSC, ASC, or STC order, the TC memory element is set to "1". In this event the sSR selector

is set to QAsSR and ∇ is set to "1" in preparation for an interchange and down right shift into A. If store clear type orders are not being executed, TC is set to "0" so OAsSR is the requested state of the sSR mechanism and ∇ is set to "0". This allows α to be placed in R and zero to be placed in S during S4. A zero must be placed in S when STN is the order. The MsS mechanism is always set to OMsS to accommodate the terminal steps of store clear type orders.

The activity in the EAU during S1 consists of placing the accumulator exponent in EA and setting sD and C such that d = ea. The delay provided by the KgA request gives the exponent decoder -- EDM (D-1504) -- time to inspect d and decide whether $d \leq -64$.

During S4, α (rounded or unrounded) is placed in R while Q or zero is placed in S depending on whether TC is set to "1" or "0". The D-adder output is gated into ES and the F-elements of the ED register are set to their new values according to the output of the exponent decoder. In particular, the $ES \leq -64$ F-element is set to "1" if $d \leq -64$. An artificial delay is also used to prevent S5 from becoming active until the output of the R zero detector (D-1505), RZ, is reliable.

In describing the S5 step, we first assume STN is the order being executed. X is set to "1", so the contents of R are transferred to M. Since $\overline{M}sS$ is requested and S contains zero, $\sigma = -m = -r = -\alpha$ where α represents the assimilated contents of A (rounded or unrounded) during S1.

Control passes to S6 where σ is gated straight into A, and zero is placed in Q. KgA is again requested simply to allow time for the A-adder output to reflect the new assimilated contents of A (new α). X is set to "0" to prevent re-entering S6 again when control passes through S4 and S5 the second time.

When S4 becomes active the second time, (new α) = -(old α) is placed

in R while zero is again placed in S. The value of d has not changed, but gES gates it into ES again. After a delay to allow RZ to become reliable, S5 is activated for the second time.

Since X is set to "0", control may pass to S9 directly if fractional overflow, normalization, or zero correction are unnecessary. Otherwise, control passes to S7 and S8 where the necessary corrections are made. We now investigate the nature of these corrections.

N is set to "1" during G1 for the SSC, ASC, STR, XCH, STC, and STN orders. This means the fractional part of the operand being stored must lie in the normalized range. For each of the above orders, the accumulator was normalized by the (R) sequence prior to the S1 entry. However, the roundoff in S1 may cause the S5 contents of R to equal $-1/4$ or $+1$ both of which are outside the normalized range.

Of those orders which enter (S) through S1 and round the contents of A to R, J is set to "0" for STU, SSC, ASC, STR, XCH, STC, and STN. This means the fractional operand being stored must not be overflowed. Note that if the fraction is guaranteed to be normalized when stored, it is also guaranteed to be non-overflowed. The converse of this is not always true. Of the above orders, only STU does not guarantee that the fraction stored is normalized.

Under all circumstances, the A-adder output, α, and the contents of R agree during S5. As discussed in section 3.1.9, $n\alpha = 0$ if $\alpha = -1/4$ while $n\alpha = 1$ if $\alpha = +1$. If $-1 \leq \alpha < 1$, $\alpha ov = 0$, but if $\alpha = +1$, $\alpha ov = 1$. Therefore, the condition $\alpha ov = 1$ must also be used to detect when α is not normalized.

It is clear that if $j = 0$ and $\alpha = +1$, the fractional overflow in R must be corrected by an effective right shift of one base 4 position and the exponent in ES (also contained by EA) increased by a unit to compensate for this shift. In this case, $\overline{j}(\alpha ov) = 1$ and $n(n\alpha) \neq 0$, so C is set to "1" and the sD

mechanism is left in the OsD state, yielding d = ea + 1 as required. The KgA request provides sufficient delay for the output of exponent decoder -- particularly the ($d \leq -64$) signal -- to become reliable. The fraction in R is changed to + 1/4 by the logic between the high end of R and FO. (See section 3.1.2.14)

If n = 1 and $\alpha = -1/4$, the fraction in R must be normalized by an effective left shift of one base 4 position and the exponent in ES (also contained by EA) decreased by unit to compensate for this shift. Since $n(n\alpha) = 1$ and $\bar{j}(\alpha ov) = 0$, C is set to "1" and -2sD is requested to yield d = ea - 1 as required. KgA is again requested to allow time for the ($d \leq -64$) output of the exponent decoder to become reliable.

When $n(n\alpha) = 1$ or $\bar{j}(\alpha ov) = 1$, it may also happen that $\bar{\lambda}[z \vee RZ \vee (es \leq -64)] = 1$. This does not affect the requests discussed above. If $\bar{\lambda}[z \vee RZ \vee (es \leq -64)] = 1$ and $n(n\alpha) = \bar{j}(\alpha ov) = 0$, the exponent is left unchanged in S5 (i.e., C is set to "0" while the sD and MsA mechanisms are bypassed).

During S7, $\bar{\lambda}[z \vee RZ \vee (d \leq -64)]$ is examined. If this expression has a value of "1" (i.e., the condition is true), the number stored in FO during S9 is considered zero. Therefore, ES is cleared to -64 and R is cleared to zero. The ($ES \leq -64$) F-element in the ED register is cleared to "1" to denote the change in ES. Note that Z is not set to "1", but it may already be in that state. There are two other interesting observations. If es = -64 during S5 with $\lambda = n(n\alpha) = 0$ and $\bar{j}(\alpha ov) = 1$, d = -63 during S7, so ES is not cleared to -64. On the other hand if es = -63 during S5 with $\lambda = \bar{j}(\alpha ov) = 0$ and $n(n\alpha) = 1$, d = -64 during S7, so ES is cleared to -64. In any event, if $\bar{\lambda}[z \vee RZ \vee (d \leq -64)] = 0$ at the time S7 becomes active, the output of the D-adder (d = ea, d = ea + 1, or d = ea -1) is gated into ES.

The S8 step is used only to check that $(\overline{OUTFL})d = 1$ before S9 becomes active. This signal is "1" when AC has unloaded the previous contents of the

FO (OUT) and set OUTFL to "0" so S9 is free to load FO with bits from R and ES.

Returning to S5 momentarily, we note that the correction path is never taken by the SAM order. SAM sets N to "0" and Λ to "1" during G1. Since it never rounds the contents of A during S1, $\alpha ov = 0$ so $\overline{j}(\alpha ov) = 0$. Consequently, SAM simply uses S5 as a null step to check that $\overline{(OUTFL)d} = 1$ before activating S9. Except for STN, the other orders which use S1 also use S5 in the same way if fractional overflow, normalization, or zero correction is unnecessary.

Before considering the action during S9, we return to the S2 and S3 entries.

SAL and SEX enter S2 directly from decode. SAL stores $0.q_{-1}q_0 \ldots q_{42}$ in FO_0 through FO_{44} and es_6, $es_5 \ldots es_0$ in FO_{45} through FO_{51}. SEX stores $0.q_{-1}q_0 \ldots q_{36}$ in FO_0 through FO_{38} and es_7, es_7, es_7, es_7, es_7, es_7, es_6, $es_5 \ldots es_0$ in FO_{39} through FO_{51}.

In S2, the contents of Q are shifted right into R with "0" placed in R_{-1} and R_0 since ∇ is set to "1". This is described in section 3.1.2.12. The exponent of AQ appears at the output of the D-adder and is gated into ES. Control then passes to S3 which is used as a null step to check that $\overline{(OUTFL)d} = 1$ prior to activating S9.

The SRM order enters S3 directly from decode. It is used primarily to store the remainder immediately following division. The (D) sequence leaves the fractional part of the remainder in R and its exponent in ES, so SRM needs only to check that $\overline{(OUTFL)d} = 1$ prior to activating S9.

During S9, RESgFO transfers the bit configurations in R_0 through R_{44} and ES_6 through ES_0 or a modified version of this configuration into FO_0 through FO_{51}. For SEX, X is set to "1" so the logic discussed in section 3.1.2.14 is used to copy es_7 into FO_{39} through FO_{44}. OUTFL and RO are set to "0" even though the latter may already be in the "0" state. OV is set to "1"

if $j(\alpha ov) = 1$, i.e., if α is overflowed and either STF or SEQ are being executed. Trouble may arise here because RO is set to "0" at the same time. Since SIF and SIA never round the contents of A in S1, $\alpha ov = 0$ for these orders even though $j = 1$. For SRM, $j(\alpha ov) = 0$ because α is never overflowed at the start of G1. OV is also set to "1" if $\overline{z}\ \overline{RZ}\ \overline{\lambda}(es_7 \oplus es_6) = 1$. That is, if the accumulator is nonzero, R is nonzero, and an arithmetic store order is executed (this excludes SAM, SAL and SEX), the number stored is overflowed if $es_7 \neq es_6$. This is clearly the case if $es_7 = 0$ and $es_6 = 1$ because its exponent is at least +64. If $es_7 = 1$ and $es_6 = 0$, its exponent is at most -65, so es is not overflowed. OV is bypassed instead of being set to "1" because RZ = 1 since R was cleared to zero in either S7 or D19 if es $\leq$ -64 in S9.

When S10 becomes active, control passes via a null step to B1 in case XCH is being executed, i.e., in case $n(tc)\theta_1\overline{\theta}_2 = 1$. If other non-store clear type orders are being executed, $(\overline{tc})(\overline{n} \vee \overline{\theta}_1 \vee \theta_2) = 1$, so ∇ is set to "0". This is necessary only for the SAL and SEX orders, but it is done for all orders in this category. When $\partial = 0$, the paths $s\sigma_{-3} \rightarrow A_{-1}$ and $s\sigma_{-2} \rightarrow A_0$ are closed while the paths $\alpha_{43} \rightarrow R_{-1}$ and $\alpha_{44} \rightarrow R_0$ are opened -- assuming in all cases the proper settings. After ∇ is set to "0", control passes to K1. Even though RO is set "0", α may be overflowed at this point so the (K) sequence must be entered. If STN is being executed and AQ contained -1 during S1, AQ contains +1 during S10. For SSC, ASC, and STC, TC is set to "1" during S10. During S1 $q_{-1}q_0 \ldots q_{44}$ was placed in $S_{-1}S_0 \ldots S_{44}$ with S_{-3} and S_{-2} set to the value of the round-off, ρ. This represents the round-off correction for the least significant half of the original contents of AQ. Since ∇ was set to "1" and OMsS was requested during S1, $s\sigma_{-3} = s_{-3} = \rho$ and $s\sigma_{-2} = s_{-2} = \rho$ appear at the inputs to A_{-1} and A_0 following the S10 request for 1/4SRsAQ. Thus, after gA is opened in S10, the bit configuration, $\rho,\rho,q_{-1}q_0 \ldots q_{41},q_{42}$ is placed in $A_{-1}A_0$ through $A_{43}A_{44}$.

If the round-off, ρ, was a unit, the least significant half of the original contents of AQ (contents during S1) is givan a negative sign as required. Since the exponent of the original contents of AQ is still in EA, S10 sets C to "0" and requests -22sD to produce the exponent, d = ea - 22, for the new accumulator contents.

In S11, this new exponent is placed in E while ∇ and TC are set to "0". Control then passes to S12.

During S12, Q is cleared to zero and Z is set to "1" if d = ea - 22 < -128. Otherwise, Z is bypassed. In any case, control passes to (K) where the accumulator is checked for zero. Note that α is never overflowed following a store clear type order.

5.8 The Shift Sequence (F)

Only the LRS and SRS orders enter the (F) sequence directly. The STF, SIF, SEQ, and SIA orders enter (F) via (P) which is discussed in section 5.9.

In all cases, the number of base 4 places that AQ must be shifted is equal to the output of the D adder, d, at the time F1 first becomes active. For LRS and SRS, d equals the exponent that is placed in EM during G1. For STF, SIF, SEQ, and SIA, d = ex - e, which is formed during P2, where e is the original exponent of AQ and ex = 0, 22, em during G1, or 6 respectively.

If $d \geq 0$, the shift direction is right. The shift direction is left if $d < 0$. When shifting right, the sign bit of the assimilated accumulator is inserted at the left end of A and S provided Λ is set to "0". Zeros are inserted at the left end of A and S if Λ is set to "1" (i.e., if the logical orders LRS and SRS are being executed). When shifting left, bits are lost at the left end of A and S, and OV is set to "1" if STF, SIF, SEQ, or SIA is the order

being performed. OV is bypassed in LRS and SRS orders since they are logical in nature.

Assume the F1 control step has just been activated by an entry from G1 or P3 as shown on D-1274. In either case, KgA = 1. The shift loop exit Eccles-Jordan, EXF, is set to "0" during G1 or P3, but the loop may not be used. It is used if $(\bar{z} \vee \lambda)(-45d44)(\overline{d = 0}) = 1$. It is bypassed to F4 if $z\bar{\lambda} \vee (d = 0) = 1$. An exit to F5 via a null step occurs if $(\bar{z} \vee \lambda)(-45d44) = 1$. These conditions are mutually exclusive and yield "1" identically when joined. We consider them separately in the order given.

Suppose $(\bar{z} \vee \lambda)(-45d44)(\overline{d = 0}) = 1$. First, this condition requires that Z be set to "0" or that an LRS or SRS is the order if Z is set to "1". Second, the shift must be in the range: $45 \leq d \leq 44$. Third, d must not be 0. The logic which generates the -45d44 signal is part of the exponent decoder which is described in section 3.2.4 and shown on EDM (D-1504). If d = -45, the left shift moves the bits in Q_{43} and Q_{44} to A_{-1} and A_0, and places zeros in all positions to the right of A_0. The AQ register could just as well have been cleared to zero in this case, since Q_{43} and Q_{44} are always cleared to zero during G1. The limit d = -45 is also used in decoding the fa #2 signal, so it was convenient to use it for controlling the shift sequence. If d = 44, the original assimilated sign bits, α_{-1} and α_0, are placed in Q_{41} and Q_{42}. There are no provisions for shortening these extreme shift operations for an SRS order. If d = 0, the entire shift loop is bypassed to F4. To summarize, F1 requests the various link mechanisms associated with the shift loop operation when $(\bar{z} \vee \lambda) = 1$ and $-45 \leq d \leq -1$ or $1 \leq d \leq 44$.

The F1, F2 loop operates very much like the A9, A10 loop. During the first pass through F1 and F2, the shift paths in the MAU are established as well as a count loop between the ES and EA registers in the EAU. if $d \leq -2$

during the first pass through F1, 2sD is requested in F2. If d = -1 or d = 1 during F1, then OsD is requested in F2. If $d \geq 2$ during F1, -2sD is requested during F2. If SRS is being executed, DL is set to "0" so only gS and gA are opened during F1 and F2. In this case, bits are either lost or zeros inserted at the right end of S and A while the contents of R and Q remain unchanged.

As an example, assume d = -6 at the time F1 first becomes active. The sEA mechanism is set to EsEA during decode and is unchanged at the time of entry to (F) even via (P). Thus $\overline{\text{ESsEA}} = 1$, so 4AQsSR and OMsS are requested while Ω is set to "1" just as in the first A9 step of the (A) loop. Opening gS and gR (provided $d\ell = 1$) causes the assimilated output of the A adder (KgA = 1) and the contents of Q to be shifted left into SR. If Λ is set to "0" and $n\alpha = 1$ (see section 3.1.9), OV' is set to "1". That is, if α is normalized at this point, a left shift into S produces modulo 2 overflow as a minimum, so OV' is set to indicate this. At the same time, d = -6 is placed in ES.

During the following F2 step, $\Omega = 1$ and $es \leq -2$ so 4SRsAQ, 2sD, KgA, and ESsEA are requested. Opening gA and gQ (provided $d\ell = 1$) causes the assimilated output of the S adder (no stored carries in S) and the contents of R to be shifted left into AQ. KgA is requested again in lieu of setting OMsA. When gEA is opened and C set to "0", the new states of the sD and sEA mechanisms cause d = -4. If

$$\text{"done"} = (es = 2) \vee (es = 1) \vee (es = -1) \vee (es = -2) \vee SZ(RZ \vee \overline{d\ell}) = 1$$

EXF is set to "1" so that the next control step is F3 instead of F1. In our example, "done" could only equal "1" if both S and R contained zero or if S contained zero and DL is set to "0". This possibility must be checked during every cycle of the shift loop regardless of the shift direction. We assume

"done" = 0 on this pass through F2. If OV' was set to "1" during F1 or if $\lambda(ns)(es \leq -2) = 1$, then OV is set to "1". Clearly, if $es \leq -2$ and the contents of S are normalized, the left shift into A causes modulo 2 overflow at least. Since A and S now contain assimilated representations, the na and ns signals are used to detect imminent overflow during the remaining cycles through the loop.

During the next pass through F1, AQ is again shifted left into SR, and d = -4 is placed in ES. Ω is set to "0" while the sSR and MsS mechanisms are bypassed to obtain fast reply signals. OV' is set to "1" if necessary.

In the following F2 step, SR is shifted left into AQ, and es = -4 is placed in EA to produce d = -2. Since $\Omega = 0$ and es = -4, the sAQ, sD, MsA, and sEA mechanisms are bypassed to obtain fast reply signals. OV is set to "1" if necessary. Unless $SZ(RZ \vee \overline{d\ell}) = 1$, EXF is again set to "0" and control returns to F1.

During this third pass through F1, the operations performed during the preceding pass through F1 are repeated. OV' is set to "1" if necessary.

During the following F2 step, SR is again shifted left into AQ by opening gA and gQ while sAQ and MsA are bypassed. In the EAU, es = -2 is placed in EA, causing d = 0. If d had been -5 originally, then es would equal -1 at this point. The sAQ and sD mechanisms would then be set to SRsAQ and OsD so that SR could be gated straight into AQ. Whether d = -6 or -5 originally, "done" is "1" at this point, so EXF is set to "1" meaning that F3 is active next in place of F1. If es = -2 and $\overline{\lambda}(ns) = 1$, OV is set to "1" for the reason given above. If es = -1 and $\overline{\lambda}(ns) = 1$, OV is bypassed since the output of SR is gated straight into AQ and overflow does not occur.

In the F3 step, DL is set to "1" to permit inspection of both A and Q for zero in K2 if necessary. The $a_0 gA_{-1}$ gate insures that AQ is not overflowed

modulo 2 when entering either the (K) or (S) sequences. Note that if $a_{-1} \neq a_0$ at the beginning of the F3 step, OV was set to "1" during the last pass through F2. When these operations are accomplished, control passes to F4.

F4 is a null step that may be entered from F1, F3, F8, or F9. Its Eccles-Jordan is set to "1" only when F4 is entered from F3. In this case, the null step acts as a delay to insure that the output of the round-off logic is reliable if R0 is set to "1" during S1.

We now return to the initial entry at F1 and assume $z\bar{\lambda} \vee (d = 0) = 1$. In this case, the loop is bypassed and control passes to F4 immediately. This occurs for the STF, SIF, SEQ, and SIA orders if Z is set to "1". There is no provision for bypassing F1, F2 if LRS or SRS is the order and the accumulator is zero (i.e., Z is set to "1"). The shift loop is bypassed for all six orders which use (F) if d = 0 at the time F1 is initially activated.

If $(\bar{z} \vee \lambda)(\overline{-45d44}) = 1$ when F1 is first entered from G1 or P3, control passes to F5 via a null step. This null step was necessary in the original design, but is not needed now. The purpose of this special bypass is to place a field of all zeros or units in A or AQ directly instead of using 46 left shifts or 45 right shifts.

If $\bar{\lambda}(d \geq 0)(\text{Sign A}) = 1$, an arithmetic right shift of more than 44 places is required, and the contents of AQ or A has a negative sign. Consequently, a field of units must be placed in AQ or A depending on whether $d\ell = 1$ or $d\ell = 0$. Therefore, F5 requests $\bar{M}MsS, \overline{CS}$ and bypasses MsS. Since the accumulator is not overflowed by a right shift, OV is bypassed.

If $\lambda \vee (d \geq 0)(\overline{\text{Sign A}}) = 1$, either a logical shift of more than 44 places is required, or else an arithmetic right shift of more than 44 places with a positive accumulator is required. In either case, a field of zeros must be placed in AQ or A depending on whether $d\ell = 1$ or $d\ell = 0$. OMsS is requested

while $\overline{\text{MMsS}}$ and CS are bypassed. AQ is not overflowed, so OV is bypassed. In terms of reply circuitry, it would have been more economical to request KMsS instead of OMsS. This would have avoided the need to bypass MsS, $\overline{\text{MMsS}}$ and CS in the three options of F5.

If $\overline{\lambda(\overline{d \geq 0})} = 1$, an arithmetic left shift of more than 45 places is required. The accumulator overflows so OV is set to "1". If AQ had been zero, the F1-F4 bypass would have occurred and F5 would not be active. Preparatory to placing a field of zeros in AQ or A, OMsS is requested while $\overline{\text{MMsS}}$ and CS are bypassed. Here again, KMsS should have been used in place of OMsS.

During F6, the output of the S-adder is gated straight into A while Q is left unchanged. Since S contains zero, $\sigma = t$, i.e., σ is the output of the MsS selector.

In F7, the contents of A, which are assimilated and either all zeros or all units, appear at the output of the A adder unaltered because KgA = 1. Thus, gR transfers α into R. KMsS is requested to clear the MsS mechanism to the OMsS state as explained in section 4.2.

As shown on D-1275, control passes to F8. If $d\ell = 0$, the S1 or K1 Eccles-Jordan is set to "1" via F4, although the F4 Eccles-Jordan remains at "0" as shown on D-1274. If $d\ell = 1$, the new contents of R, which agree with the present contents of A, are gated straight into Q and control passes to F9.

F9 is a null step in which the Eccles-Jordan in S1 or K1 is set to "1" via F4, although the F4 Eccles-Jordan remains at "0". This null step acts as a delay to insure that the output of the round-off logic is reliable if RO is set to "1" during S1.

5.9 The Store Preliminaries Sequence (P)

The logic for this sequence appears on D-1275. The (P) sequence is

used by the STF, SIF, SEQ, and SIA orders. The purpose of this sequence is to establish the number of base 4 places the accumulator must be shifted before its exponent equals the one specified by STF (e = 0), SIF (e = 22), SEQ (e = em), or SIA (e = 6).

At the start of each of these orders, G1 transfers the exponent of AQ to EA from E and sets OsD and $\overline{C}$. Therefore, when P1 becomes active, the exponent of AQ appears at the output of the exponent adder, and the exponent of what was in F1 (IN) during G1 is in EM.

During P1, d is placed in ES and E is loaded as follows:

STF	$0 \rightarrow E$	SEQ	$em \rightarrow E$
SIF	$22 \rightarrow E$	SIA	$6 \rightarrow E$

If the exponent decoder finds that d = 0 when STF is the order being executed, i.e., if $\overline{\theta}_1\overline{\theta}_2(d = 0) = 1$, control passes to S1 directly. Otherwise, the P2 Eccles-Jordan is set to "1".

In P2, the exponent of AQ is transferred from ES to EM, and the contents of E are placed in EA. By setting $\overline{EM}$sD and C to "1", the new value of d may be expressed as follows where em represents the present exponent of AQ and ea represents the contents of EM during P1.

STF	$d = 0 - em$	SEQ	$d = ea - em$
SIF	$d = 22 - em$	SIA	$d = 6 - em$

It is clear that d is the shift difference which is used by the (F) sequence as explained in the preceding section.

During P3, the EXF Eccles-Jordan is set to "0" since control now passes to F1. Note that the actual request for $\overline{EXF}$ appears on D-1274.

5.10 The Normalize Sequence (R)

The (R) sequence logic appears on D-1282. It is the first sequence after G1 for MPY, DIV, NDV, VID, STR, XCH, STC, STN, and DAV. (R) is used to "normalize" the fraction, f, in AQ or A depending on whether DL is set to "1" or "0". Except for MPY and DAV, the (R) sequence left shifts the fraction in AQ or A until it is normalized (i.e., $-1 \leq f < -1/4$, $f = 0$, or $1/4 \leq f < 1$). In the case of MPY or DAV, the left shift stops when the fraction in AQ is normalized or when Q contains zero, whichever occurs first. For all cases except the re-entry from D4, DL is set to "1" so the contents of AQ are normalized. When (R) is entered from D4, the fraction in A is normalized while the contents of Q are left unchanged.

All entries to (R) gate the EXR status memory element which controls exit from the loop (R1, R2). The positive logic for setting EXR to "1" (meaning exit from the normalize loop) is given below.

$$EXR = n\alpha \vee z(d\ell) \vee (za)(\overline{d\ell}) \vee \bar{n}\,\bar{\alpha}_1\,\bar{\theta}_2(QZ) \vee j(QZ)$$

It is important to note that if EXR is set to "1" during entry to (R), then R3 instead of R1 becomes active -- meaning the normalize loop is bypassed entirely. This occurs if α is normalized (KgA = 1 so α is the assimilated representation of A), if Z is set to "1" initially, or if MPY or DAV is the order and Q contains zero, i.e., QZ = 1. Since ZA is always set to "0" during G1, an immediate bypass of (R) when entering from D4 is not possible. However, if A is found to contain zero during any pass through R2, then ZA is set to "1" and exit occurs. Because the exponent of AQ or A is reduced by a unit for each base 4 left shift of the fraction, underflow (i.e., $d < -128$) may occur and set Z to "1" during any pass through R2. In the case of MPY or DAV, Q may become zero during any pass through R2, causing an exit from the normalize loop even though the contents of AQ may not be normalized.

Assume that EXR is set to "0" during the initial entry to R1. The fraction to be "normalized" is in AQ or A and its exponent is in E. KgA = 1, so α is the assimilated representation of A. The assimilated contents of AQ or A is shifted left into SR or S depending on whether DL is set to "1" or "0". To allow the new contents of S to appear at the output of the S-adder as σ, the MsS mechanism is set to the zero state. The exponent in E, e, is placed in EA, and the sD mechanism is set to -2sD. If α will appear normalized after a single base 4 left shift, $n\alpha' = 1$. Otherwise, $n\alpha' = 0$ as discussed in section 3.1.9. If the new contents of SR or S are normalized, $n\alpha' = 1$ and the exponent should only be decreased by a unit. Thus, $n\alpha' = 1$ sets C to "1", leaving d = e - 1. If the new contents of SR or S are not normalized, $n\alpha' = 0$ and at least one more base 4 left shift will be required, so the exponent should be decreased by two units. Therefore, $n\alpha' = 0$ sets C to "0", leaving d = e - 2.

During R2, the contents of SR or S (now in assimilated form) are gated left or straight into AQ depending on whether C is set to "0" or "1" as explained above. The output of the exponent adder, d = e - 1 or d = e - 2 is placed in E. If $d < -128$, dz = 1 so Z is set to "1", indicating that the accumulator is now considered zero. If S is found to contain zero, sZ = 1 so ZA is set to "1" provided DL is set to "0". That is, if S contains zero during this first pass through R2 and a single length normalization is being performed ((R) entered from D4), the number being normalized is zero. Furthermore, if R is gated into Q, the contents of Q may now be zero in which case QZ = 1. EXR is set to "0" or "1" during R2 depending on the output of the logic given above.

If exr = 0, R becomes active again, and the loop continues until EXR is set to "1". If exr = 1, R3 is activated instead of R1, and control passes to S1, V1, M1, or D1 depending on the order. For STR, XCH, STC, and STN, $n\bar{j} = 1$ so s1 is activated. For DAV, j = 1, so V1 is activated. For MPY, $\bar{n}\,\bar{\theta}_1\bar{\theta}_2 = 1$, so M1 is activated. For DIV, NDV, and VID, $\bar{n}\,\bar{j}(\theta_1 \vee \theta_2) = 1$, so D1 is activated.

5.11 The Difference Absolute Value Sequence (V)

The logic for this sequence is shown on the right side of D-1280. It is used only by the DAV order as indicated in Table I.

When V1 is activated following the (R) sequence, the accumulator is either normalized or at least Q contains zero. During the first pass through V2, V3, and V4, the contents of A are rounded and placed in M, while the absolute value of the contents of M is placed in A with Q cleared to zero. During the second pass through V2, V3, and V4, this process is repeated, and θ_1 is set to "0" so that $a_R|-|m|$ is formed by the (A) sequence.

When V1 is activated following (R), α represents the rounded value of the contents of Q. The exponent of the accumulator is placed in EA while the sD mechanism is set to $\overline{EM}$sD and C is set to "0". These operations establish the initial state of the EAU prior to entering the (A) sequence.

During V2, α is placed in R while zero is placed in S. MsS is requested if M contains a positive number; $\overline{M}$sS is requested if M contains a negative number. Thus $\sigma = |m|$.

In V3, $|m|$ is placed in A while Q is cleared to zero. RO is set to "0" since round-off is complete. Since j = 1, θ_1 is set to "1" to obtain a "0" θ_1r signal even though θ_1 is already in the "1" state. This avoids a bypass request to θ_1.

During V4, the contents of R (a_R) are placed in M. J is set to "0" so that when V3 becomes active the next time, θ_1 will be set to "0". EXA is set to "0" as a preparatory step to entering the (A) sequence.

Since $\theta_1 = 1$ following the first pass through V4, control returns to V2. At this point, a_R appears in M and $|m|$ appears in A. During V2, $|m|$ is placed in R and zero is again placed in S. If a_R is positive, MsS is requested. If a_R is negative, $\overline{M}$sS is requested. Therefore, $\sigma = |a_R|$.

In V3, $|a_R|$ is placed in A while Q is again cleared to zero. RO is again set to "0" simply to obtain a zero reply. During this pass through V3, j = 0, so θ_1 is set to "0".

In V4, |m| is transferred to M. J and EXA are both set to "0" again to obtain "0" replies. Because θ_1 = 0 at this point, control passes to A1. Note that θ_1 = 0 and the other status memory element settings for DAV cause a subtraction to occur during the (A) sequence. The exponents of |m| and $|a_R|$ are in EM and EA respectively. Since $\overline{EM}$sD and $\overline{C}$ were requested during V1, d = ea - em - 1 appears at the output of the D-adder when A1 becomes active.

It is interesting to note that $|m| = |a_R| = +1$ is a possible result of the (V) sequence. The logic at the most significant end of the A register handles this case correctly during the (A) sequence.

If $|m| \neq D$ and $|a_R| = 0$ when A1 becomes active, control passes direc-16 to B1 as discussed in section 5.2. The (B) sequence performs a CSB in the usual manner.

5.12 The Multiply Sequence (M)

The logic for this sequence is shown on D-1281. It is used following (R) in the execution of an MPY order as shown in Table I.

At the time M1 becomes active, the unrounded multiplier is in AQ and E. The fraction in AQ is normalized or at least Q contains zero. The multiplicand resides in M and EM. If the multiplier is zero, Z is set to "1". If the multiplicand is zero, em = -64.

Let the rounded multiplier, which may or may not be normalized, be represented by $a_R \cdot 4^e$. Let the multiplicand be represented by $m \cdot 4^{em}$. The exponent of the product is formed in the EAU as e + em while the 90 bit fractional product is formed in the MAU as $a_R \cdot m$. Thus $p \cdot 4^e = (a_R \cdot m) \cdot 4^{(e + m)}$.

As explained in section 3.1.15, the 46 bits fractional multiplier (including two sign bits) is recoded two bits at a time so that 2m, m, 0m, and -m are the required multipliers of the fractional multiplicand. The 90 bit fractional product (including two sign bits) is obtained with 23 additions and 22 right shifts. The first two additions and right shifts occur during M3 and M4. The next 20 occur during M5 and M6 which comprise the multiply loop. The final addition without shift into AQ occurs during M7.

As in section 3.1.15, assume each two bits of the rounded multiplier represents a base 4 multiplier digit β_i so that $\alpha_{-1}\alpha_0$, $\alpha_1\alpha_2$ ... $\alpha_{43}\alpha_{44}$ becomes β_0, β_1 ... β_{22} where $\beta_i = 0, 1, 2, 3$. The recoded base 4 digits B_i' are defined as follows:

$$\beta_i' = \beta_i + \lambda_i - 4\lambda_{i-1}$$

where $\lambda_{-1} = \lambda_{22} = 0$ and $\lambda_i = 0$ or 1 for $0 \leq i \leq 21$.

The shifted partial products, $\frac{p_i}{4}$, held in AQ and SR at each step of the multiplication are listed below in terms of β_i'.

M3 AQ: $$\frac{p_1}{4} = 1/4 \left[\beta_{22}' m\right] = \left[\frac{\beta_{22}'}{4}\right] m$$

M4 SR: $$\frac{p_2}{4} = 1/4 \left[\frac{p_1}{4} + \beta_{21}' m\right] = \left[\frac{\beta_{21}'}{4} + \frac{\beta_{22}'}{4^2}\right] m$$

M5(1st) AQ: $$\frac{p_3}{4} = \left[\frac{\beta_{20}'}{4} + \frac{\beta_{21}'}{4^2} + \frac{\beta_{22}'}{4^3}\right] m$$

M6(1st) SR: $$\frac{p_4}{4} = \left[\frac{\beta_{19}'}{4} + \frac{\beta_{20}'}{4^2} + \frac{\beta_{21}'}{4^3} + \frac{\beta_{22}'}{4^4}\right] m$$

M5(2nd) AQ: $\frac{p_5}{4} = \left[- - - - - - - \right]m$

M6(2nd) SR: $\frac{p_6}{4} = \left[- - - - - - - \right]m$

- -

- -

M6(10th) SR: $\frac{p_{22}}{4} = \left[\frac{\beta'_1}{4} + \frac{\beta'_2}{4^2} + \ldots + \frac{\beta'_{22}}{4^{22}}\right]m$

M7 AQ: $p = p_{23} = \left[\beta'_0 + \frac{\beta'_1}{4} + \frac{\beta'_2}{4^2} + \ldots + \frac{\beta'_{22}}{4^{22}}\right]m$

During M1, control passes to M8 via a null step if $(em = -64) \vee z = 1$. This null step is not necessary in the present control design.

In M8, Z is set to "1", indicating a zero product. OV is bypassed simply to obtain a zero reply and thus set the G1 Eccles-Jordan to "1".

If $\overline{z(em = -64)} = 1$ during M1, the contents of A are rounded by setting R0 to "1" and requesting KgA. The assimilated, rounded multiplier appears as α. The μ status memory element is set to "1" to permit the mode bits of the recoded multiplier to be gated into R^*_{42} and Q^*_{42} as indicated in section 3.1.15. In the EAU the exponent of the product appears as d = em + ea = em + e = ep.

During M2, the two least significant bits of the rounded multiplier, $\alpha_{43}\alpha_{44}$, are recoded to determine the setting of the MsS mechanism. At the same time, α is placed in R and zero is placed in S. The mode bit that is generated as a consequence of recoding $\alpha_{43}\alpha_{44}$ is also placed in R^*_{42}. This bit influences

the recoding of the next base 4 multiplier digit which is presently held in R_{41} and R_{42}. Note that the recoded version of α_{43} and α_{44} is not stored in R but is simply reflected in the setting of the MsS mechanism. This is true for all steps of the multiplication process. In the EAU, ep is placed in E and the ESOV and ESZ F-elements are set to "1" if necessary. The count of 21 is selected as an input to EA but is not actually gated into EA until M3.

In M3, 2m, m, 0m, or -m is added to 0 in S to form the first partial product, p_1, which is then shifted right into AQ. At the same time, r_{41}, r_{42}, and r^*_{42} are recoded by the μMsA logic which sets the MsA mechanism accordingly. The generated mode bit is placed in Q^*_{42}. The R0 memeory element is set to "0" as a clean-up operation. In the EAU, 21 is transferred to EA and -2sD is selected so that d = 19. If esov = 1, 0V is set to "1" to indicate ep > 127. If esz = 1, Z is set to "1" to indicate ep < -128. Otherwise, 0V and Z are bypassed.

During M4, 2m, m, 0m, or -m is added to $\frac{p_1}{4}$ in AQ to form p_2 which is then shifted right into SR. At the same time, q_{41}, q_{42}, and q^*_{42} are recoded by the μMsS logic which sets the MsS mechanism accordingly. Tne new mode bit is placed in R^*_{42}. In the EAU, 19 is placed in ES while the output of ES is selected as the input to EA.

During each of the ten passes through M5, the new partial product, which appears as σ and the most significant portion of R, is shifted right into AQ. The next base 4 multiplier digit, $r_{41}r_{42}$, along with the associated mode bit, r^*_{42}, is recoded and used to set the MsA mechanism. The new mode bit is placed in Q^*_{42}. The count held in ES is gated into EA.

During each of the ten passes through M6, the new partial product, which appears as α and the most significant portion of Q, is shifted right into SR. The next base 4 multiplier digit, $q_{41}q_{42}$, together with its mode bit, q^*_{42},

is recoded and used to set the MsS mechanism. The new mode bit is placed in R^*_{42}. The count d = ea - 2 is placed in ES.

The final pass through M6 merits special attention. The sign digit of the rounded multiplier, $\alpha_{-1}\alpha_0$, is held in $Q_{41}Q_{42}$ at the start of this step. The new mode bit in Q^*_{42} may be "1" or "0". The recoded digit, β'_0, is determined as usual. If a mode bit is generated, it is thrown away, since d = -1 and μ is set to "0". Thus, a "0" is gated into R^*_{42}. Note that if the rounded multiplier is negative (i.e., $\alpha_{-1} = \alpha_0 = 1$), $\beta_0 = 3$. If $q^*_{42} = 0$, $\beta'_0 = -1$ as required. If $q^*_{42} = 1$, $\beta'_0 = 0$ again as required. In either case, the next mode bit is always "0".

It is worthwhile to observe why R^*_{42} must be set to "0" during the final pass through M6. The two least significant bits of the product (actually the last shifted partial product) are placed in R_{41} and R_{42} during this pass. In M7 the outputs of the borrow subtractor d_{41} and d_{42} are gated into Q_{41} and Q_{42} as the two least significant bits of the product. Clearly, they should be copies of r_{41} and r_{42}. However, they will not be unless R^*_{42} is set to "0" as discussed in section 3.1.17.

In M7, the final multiple of the multiplicand is added to the last shifted partial product in SR to obtain the product as σ together with the first 44 bits of R. The product is transferred straight into AQ, and KgA is selected to assimilate the stored carries in A. The assimilated fractional product, p, appears as α together with the first 44 bits of Q at the entrance to K . Note that $-1 \leq p \leq 1$ at this point.

5.13 The Division Sequence (D)

The logic for (D) is shown on D-1271, D-1272, and D-1273. It is used in conjunction with (R) and (K) to execute DIV, NDV, and VID. Note that D1, D2,

or D5 is always activated by a request from R3. Furthermore, if the divisor is not normalized or if NDV or VID is the order, the (R) sequence is re-entered from D4.

The requests made by the (D) sequence in D1, D2, D3, and D4 depend on the divide order and the range of the divisor (dividend in the case of VID). At the start of D6, a fractional divisor in the range $-1 \leq d \leq -1/4$ or $1/4 \leq d \leq 1$ appears in M while the fractional dividend in AQ lies in the range $-1 \leq D < -1/4$, $D = 0$, or $1\ 4 \leq D < 1$. The corresponding quotient exponent, eq, appears at the output of the exponent adder as $d = ea - em = eD - ed = eq$, where eD and ed represent the "normalized" ($1/4 \leq |x| \leq 1$) dividend and divisor exponents respectively.

To insure that the fractional quotient is less than or equal to unity in magnitude, the dividend D is subnormalized in D6. The subnormalized dividend D' lies in the range $-1/4 \leq D' < -1/16$, $D' = 0$, or $1/16 \leq D' < 1/4$. The fractional quotient $q' = \frac{D'}{d}$ must lie in the range $-1 \leq q' \leq -1/16$ or $1/16 \leq q' \leq 1$ where $D' = 0$ results in a premature exit to (G). The corresponding exponent correction $eq' = eq + 1$ is not made at this point but is taken into account during the terminal steps of the (D) sequence.

For all three divide orders a modified binary non-restoring division algorithm is used to generate the fractional quotient. The recursive relationships are given in section 3.1.16. The modification consists of allowing the predicted quotient "bits" to assume the values 1, 0, -1. The term "binary" still applies since the weighting associated with each "bit", y_i, is still 2^{-i}. As discussed in sections 3.1.16 and 3.1.17, the "bits" y_{2k+1} and y_{2k} ($0 \leq k \leq 22$ or 23), are generated separately and then recoded as a base 4 quotient digit, q_k, which may assume any value between 3 and -3.

The y_{2k+1} "bit" is determined by the ρMsS logic shown on D-1507. This

logic is fed by m_{-1} and the most significant outputs of the A-adder. The setting of the MsS mechanism is also governed by this logic. If $y_{2k+1} = -1$, $\overline{2M}sS$ is selected, while $y_{2k+1} = 0$ or 1 cause 0MsS or 2MsS to be selected.

The y_{2k} "bit" is determined by the ρMsA logic which also appears on D-1507. This logic is fed by m_{-1} and the most significant outputs of the S-adder. It also determines the setting of the MsA mechanism. The "bits" $y_{2k} = -1$, 0, and 1 correspond to settings of $\overline{M}sA$, 0MsA, and MsA.

In D6, the 0MsS setting and gH_1' correspond to $y_{-1} = 0$ for all d and D'. The value of y_0 and the corresponding setting of the MsA mechanism are determined during the following straight transfer into AQ with D'. The partial remainders (shifted or unshifted) that appear at each step of the division process are given below.

D7: AQ: $R_0 = D' - 0d$

D9: SR: $4R_1 = r(R_0 - y_0 d) = 4(D' - y_0 d)$

D10(1st pass): AQ: $R_2 = 4R_1 - 2y_1 d = 4D' - d(4y_0 + 2y_1)$

D11(1st pass): SR: $4R_3 = 2^4 D' - d(2^4 y_0 + 2^3 y_1 + 2^2 y_2)$

D10(2nd pass): AQ: $R_4 = 2^4 D' - d(2^4 y_0 + 2^3 y_1 + 2^2 y_2 + 2y_3)$

D11(2nd pass): SR: $4R_5 = 2^6 D' - d(2^6 y_0 + 2^5 y_1 + 2^4 y_2 + 2^3 y_3 + 2^2 y_4)$

- -

- -

D10(22nd pass): AQ: $R_{44} = 2^{44} D' - d(2^{44} y_0 + 2^{43} y_1 + \ldots + 2y_{43})$

D11(22nd pass): SR: $4R_{45} = 2^{46} D' - d(2^{46} y_0 + 2^{45} y_1 + \ldots + 2^2 y_{44})$

The division process stops at this point if the fractional quotient, $q' = y_0.y_1y_2 \ldots y_{44}$, appears "normalized" as discussed in section 3.1.10. Thus, if nr = 1, the remainder associated with the unrounded quotient is $2^{-44}R_{45} = D' - d(y_0 + 2^{-1}y_1 + \ldots + 2^{-44}y_{44})$, so that $D' = dq' + 2^{-44}R_{45}$. If nr + 0, the division process is allowed to proceed two more steps to obtain a normalized quotient.

D10(23rd) AQ: $R_{46} = 2^{46}D' - d(2^{46}y_0 + 2^{45}y_1 + \ldots + 2y_{45})$

D11(23rd) SR: $4R_{47} = 2^{48}D' - d(2^{48}y_0 + 2^{47}y_1 + \ldots + 2^2y_{46})$

The "bits" $y_{-1} = 0$ and y_0 are discarded so that $q = y_1y_2.y_3y_4 \ldots y_{46}$ becomes the "normalized" fractional quotient with $2^{-44}R_{47} = 4D' - m(2y_1 + y_2 + 2^{-1}y_{46})$ as the remainder, or $4D' = dq + 2^{-44}R_{47}$. Note that q' or q denotes the fractional quotient depending on whether it appears "normalized" after 23 or 24 divide loop cycles.

In either case one more quotient "bit" $y_{45}(y_{47})$ is always determined during the final pass through D10. This "bit" establishes the MsS setting during D11. Furthermore, $y_{45}(y_{47})$ and a sign comparison of divisor and next partial remainder (partial remainder beyond the final remainder) is used to round the quotient positively, negatively, or not at all. The rules for round-off are given below. Refer to section 3.1.16 for the coding of $y_{45}(y_{47})$.

$y_{45}(y_{47})$	$m_{-1} + \alpha_{-1} = cb$	Quotient Round-Off
-1	0	0.2^{-44}
-1	1	-1.2^{-44}
0	0	0.2^{-44}
0	1	0.2^{-44}
1	0	$+1.2^{-44}$
1	1	0.2^{-44}

The quotient round-off is determined while the shifted final remainder is in S. Therefore, the final remainder, $R = 2^{-44}R_{45}$ (or $2^{-44}R_{47}$), is corrected for quotient round-off before the latter is accomplished. That is, $R_r = R + m$ if the round-off is negative, $R_r = R - m$ if the round-off is positive, and $R_r = R$ if the round-off is zero. This occurs during D14 while q is in Q.

The actual round-off of the quotient occurs in D17. The quotient in Q is transferred to S and rounded appropriately during the straight gate into Q in D18.

The quotient exponent, eq, must be corrected for subnormalization of the dividend and possibly for the extra two steps in the divide loop (D10 and D11). This correction is made during D12 and D13. The corrected quotient exponent, eq', is placed in E during D14. The appropriate remainder exponent, eR, is derived in D15, D16, and D17. It is placed in ES during D19 unless $eR \leq -64$ in which case -64 is placed in ES.

We now proceed to describe each step of the (D) sequence in detail. For all three divide orders, the contents of AQ are normalized before D1, D2, or D5 is activated. In discussing D1 through D5, it is convenient to consider DIV and NDV first.

Assume that a DIV or NDV order is being executed and that the contents of AQ during decode have just been normalized. On D-1271 note that R3 must activate D2 or D5 rather than D1 since $\bar{n}\bar{j}\bar{x}\theta_1\theta_2 = 0$ except for a VID order. Note ruther that D5 can only be activated when the number in M (i.e., the divisor) is normalized, nm = 1, and the order is DIV. Under these conditions, $\bar{x}(nm)\bar{\theta}_1\theta_2 = 1$. The second possibility, $x\theta_1\theta_2 = 1$, only applies for a VID order following the second exit from (R). Assume that D2 becomes active either because nm = 0 or because the order is NDV.

In D2, α is placed in R and zero is placed in S. Since KgA = 1 during

decode, α represents the assimilated contents of A. If DIV is the order, MsS is requested so that the unnormalized divisor m = σ appears at the output of the S-adder. For NDV, MsS is requested so that -m = σ. In the EAU, the exponent of the accumulator is placed in EA and reflected as d at the output of the D-adder for either DIV or NDV.

During D3, m or -m is placed in A and the gQ gate is bypassed since Q contains the least significant half of the dividend in the case of DIV or NDV order. R0 is set to "0" as a clean-up operation. Except for the VID order, R0 is already in the "0" state at the start of D3. Since X is always set to "0" during G1 for DIV, NDV and VID, DL is set to "0" during this pass through D3. In the EAU, d = eD (i.e., the exponent of the normalized dividend) is placed in ES. Before normalization or negation the exponent of the divisor, em, is placed in E.

In D4, the most significant half of the dividend is transferred from R to M. X is set to "1" to indicate the first pass through D2, D3, and D4. In the EAU, eD is transferred from ES to EM. Since D1 is set to "0", D6 cannot be activated. For DIV and NDV either θ_1 or θ_2 is set to "0" so that $d1(\bar{z} \vee \bar{\theta}_1 \vee \bar{\theta}_2) = 1$. This is the condition for setting EXR to "0" or "1" via gEXR and the input function discussed in section 5.10. EXR is set to "0" if the fractional divisor (m or -m) is not normalized. Since ZA is always set to "0" during G1, D5 cannot be activated because the A register contains a zero at this point. Hence, for DIV and NDV, a premature exit to G1 via D5 with OV set to "1" cannot occur during this pass through D2, D3, and D4. It can occur for VID if Z is set to "1" as discussed later.

The signal which opens the gate to EXR also activates R1. If EXR is set to "1", R3 becomes active and control is transferred immediately to D2 in the case of DIV or NDV. If EXR is set to "0", the normalization loop (R1, R2)

is initiated. The principal difference is that dl = 0, so only the contents of A are normalized. The other difference is that ZA is set to "1" if S contains zero during the first pass through R2. The exponent of the unnormalized divisor is decreased by a unit for each left shift of the fractional part in A. Since em > -64 initially if the fractional divisor is nonzero, the divisor must become normalized before dz = 1 (i.e., before the D-adder output d > -128). Thus Z is always bypassed during R2 in this second pass through (R). If ZA is not set to "1" during the initial pass through R2, (R) continues to normalize single length until the fraction is normalized (i.e., $n\alpha = 1$).

When control returns to D2, as it must for a DIV or NDV order, the normalized fractional divisor, d, is in A with its exponents, ed, in E. During D2 the contents of A (i.e., d) is placed in R and zero is placed in S. MsS is selected so that $\sigma = m$ which represents the most significant half of the normalized fractional dividend. The divisor exponent, ed, is transferred from E to EA and reflected at the otuput of the D-adder.

During D3, $\sigma = m$ is placed in A, but gQ is again bypassed. The normalized fractional dividend is now in AQ. RO is again set to "0" and DL is set to "1" since x = 1 during the second pass through D2, D3, and D4. In the EAU, ed is placed in ES while eD is transferred from EM to E.

During D4, the normalized divisor is transferred from R to M while ed is placed in EM. X is again set to "1". D5 is now activated.

If za = 1, meaning that the fractional divisor is zero, OV is set to "1" and control is transferred to G1. The normalized dividend may be zero in which case Z is also set to "1". Otherwise, $D \cdot 4^{eD}$ appears in AQ and E. The divisor, $d \cdot 4^{ed}$, appears in M and EM. If d = 0, ed = -66 provided em = -64 during G1.

If za = 0, $dl(\overline{za}) = 1$ and the quotient exponent, eq = eD - ed, is

formed at the output of the D-adder. This exponent may or may not be the exponent of the final quotient. It will be if $q = (D/d)$ lies in the normalized range. In this case, the effect of subnormalization in D6 (i.e., the effect of shifting D one base 4 position to the right) is exactly canceled by the extra pass through D10 and D11 to achieve a normalized quotient. If $q = (D/d)$ lies outside the normalized range (i.e., $1 < |q| \leq 4$), $eq' = eq + 1$ will be the exponent of the final quotient $q' = \frac{q}{4}$. In this case the subnormalization of D is not canceled by an extra pass through D10 and D11 since q' appears normalized in R after 23 cycles of the divide loop. The addition of zero or a unit to $eq = eD - ed$ occurs during D13 based on the count in ES after exit from the divide loop (D10, D11).

We return now to the case of a VID order. This order considers the contents of the accumulator as the divisor. Following G1, control is transferred to R. The fraction in AQ is normalized as usual with exit from (R) occurring as a consequence of $n\alpha = 1$ or $z(d1) = 1$. In contrast with DIV and NDV, R3 activates D1 since $\bar{n}\bar{j}\bar{x}\theta_1\theta_2 = 1$.

During D1, KgA is selected and R0 is set to "1" so that α represents the rounded and assimilated value of A. It is this quantity which is used as the fractional divisor. Since $x = 0$ during this entry from (R), control passes to D2, D3, and D4 where the data handling is identical to that of a DIV order. The opening of qQ during D3 to clear Q to zero is the one exception to this.

During D4, the gate to the EXR status memory element is opened and R1 is activated if Z is set to "0", i.e., if $(\overline{d1})(\bar{z}) = 1$. If $z = 1$, the quantity in AQ at the start of the VID order was determined to be zero during (R) so the divisor is zero. Hence, $z\theta_1\theta_2 = 1$ activates D5 which sets OV to "1" and exits to G1.

If $z = 0$, the R sequence is entered for the second time. At this point, M and EM contain the normalized and rounded <u>divisor</u> while the quantity

in A and E represents the single length dividend as brought in from memory during G1. Note that the fraction in M was rounded following normalization and may be +1 or - 1/4. Either of these values may be used as the fractional divisor which is why m_{-1} is used as the divisor sign bit in the predictor logic as discussed in section 3.1.16.

During this pass through (R), the fractional dividend is normalized single length even though Q contains zero. Exit occurs when $n\alpha = 1$ or $(za)(\overline{d1}) = 1$. Control is transferred from R3 to D5 since X was set to "1" during the previous pass through D4 and $\theta_1 = \theta_2 = 1$ for VID.

In D5, the quotient exponent, eq = eD - ed, is calculated as described for the DIV and NDV orders. Again, it is important to note that $eq' = eq + 1$ rather than eq may be the exponent of the fractional quotient which is actually computed.

From this point on the discussion encompasses all three divide orders.

During D5, the fractional dividend, D, is in AQ with its exponent, eD, in E and EA while the fractional divisor, d, is in M with its exponent, ed, in EM. While eq is computed in the EAU, the setting of Z is examined. If z = 0, D6 is activated. If Z = 1, the dividend is zero in the case of a DIV or NDV order, so D8 is activated.

During D8, R is cleared to zero and -64 is placed in ES so that a zero remainder is stored if an SRM order follows the divide order. OV is also bypassed in order to activate G1. This is the second premature exit from (D).

If D6 is activated following D5, the fractional dividend, D, is sub-normalized by shifting it right one base 4 position into SR. OMsS is selected so that the most significant half of $D' = D/4$ appears at the output of the S-adder. To agree with this selection, gH_1' sets the H_1' memory element in the ρMsS predictor (D-1507) to "1". The effect of this is described in section 3.1.17.

In case VID is the order being executed, dl = 0 at this point. Therefore, DL is set to "1" to permit double length left shifts during the first 23 steps of the division process. In the EAU, eq = eD - ed is placed in ES via gES which also gates the output of the exponent decoder into the ED register (D-1504). The count for the divide loop is established by placing 22 in E.

As a part of D6, a decision is made to activate D7 or D8 depending on whether $(\overline{esz})(\overline{za})$ = 1 or 0. If this expression has a value of "1", eq $\geq$ -128 and ZA is set to "0". This means the quotient exponent is not underflowed and, in the case of a VID order the dividend is not zero. The converse holds if $(esz) \vee za = 1$. In this case Z is set to "1" to indicate a zero quotient and control passes to D8.

In D8, R is cleared to zero and -64 is placed in ES indicating a zero remainder. Following D8, control is transferred to G1 as usual.

If control passes from D6 to D7, the subnormalized dividend, D', is transferred straight into AQ. The ρMsA predictor determines the setting of the MsA mechanism in the manner described in section 3.1.16. If y_0 is 1 or -1, G2 is set to "0" or "1" and H_2 is set to "0". In the EAU, 22 is transferred from E to EA. The selection of -2sD causes 21 to appear at the output of the D-adder since C was previously set to "1". Furthermore, OV is set to "1" if esov = 1 (i.e., if eq > 127). Otherwise, OV is bypassed.

At the start of D9, the most significant half of the unassimilated first partial remainder appears as α. The ρMsS predictor logic determines the setting of the MsS selector mechanism while α and Q are shifted left into SR.

The outputs of the G_1, H_1, G_2, and H_2 memory elements (D-1507) are recoded by the quotient digit recoder logic (D-1507) to form a base 4 quotient digit which may lie anywhere in the range $-1 \leq q_0 \leq 1$ during this step of the division process.

As described in section 3.1.17 and shown on D-1524, δ_1 must be "1" before ρ_{43} and ρ_{44} can be gated into R_{43} and R_{44} via gR. Inspection of D-1272 will reveal that $\delta_1 = 1$ during D9. It should also be noted on LQR (D-1524) that β_{42} cannot be set into R^*_{42} by gR unless $\delta_2 = 1$. This condition is not true during D9 so a "0" is placed in R^*_{42}. This is done to prevent the borrow-subtractor logic from decreasing the first partial remainder by a unit in the least significant position (i.e., the 42nd position) during the straight transfer into Q in D10. If $q_0 = 1$, $\beta_{42} = \rho_{43} = \rho_{44} = 1$ during D9, but a "0" is placed in R^*_{42} while "1" is placed in R_{43} and R_{44}. Since q_0 represents the unmodified base 4 sign digit of the quotient, the "1"s in R_{43} and R_{44} will eventually be interpreted as -1 even though they both have positive weight at this point.

During D9 and the rest of the division process, the EAU is used as a counter. The output of the D-adder during D9 is d = 21. This output is placed in E via DgE. The exponent of the normalized divisor, ed, is held in EM while the quotient exponent, eq = eD - ed, is held in ES. At this point eq has not been corrected in accordance with the subnormalization that occurred during D6.

Following the completion of D9, control passes to the divide loop which consists of D10 and D11. Exit to D12 depends on the count in the EAU having reached -1 and on the appearance of a "normalized" quotient in R as discussed in section 3.1.10

During every pass through D10, the new partial remainder at the output of the S-adder and in the most significant end of R is transferred straight into AQ. The partially formed quotient which resides in the remaining bits of R is transferred into the corresponding bits of Q. If $R^*_{42} = 1$, the borrow-subtractor logic reduces the quotient by a unit in the 42nd position as described in section 3.1.17. If $r^*_{42} = 1$ and $r_{43} = r_{44} = 0$, a borrow is made in anticipation of a negative quotient digit in the future. The bit which is borrowed is held as

a stored carry in Q^*_{42} since the quotient digit inserted in R^*_{42}, R_{43}, and R_{44} is in fact zero. R^*_{42} is set to "1" when $\gamma = 1$, i.e., when the true sign of the previous partial remainder in AQ and the sign of the divisor disagree. This means a negative quotient digit will eventually occur even though the ρMsS and ρMsA logic predicted a zero.

During each pass through D10, the ρMsA logic at the output of the S-adder determines the new setting of the MsA mechanism as MsA, OMsA, or MsA and sets G_2 and H_2 accordingly. Furthermore, the contents of G'_1 and H'_1 are copied into G_1 and H_1 so that the recoder can determine the next base 4 quotient digit. Note that even though this digit may be zero, β_{42} may be "1" because $\gamma = 1$. As shown on CG (D-1099), SEC (D-1527), and DPα (D-1507), γ is "1" if the true sign of the partial remainder placed in AQ disagree with the sign of the divisor; i.e., a negative quotient digit will eventually arise in the infinite quotient. Since the borrow subtractor can only effect the last two bits of the quotient, a borrow must be made on the next step in anticipation of this negative quotient digit as explained above.

During every pass through D10, Δ_2 is set to "1" so that β_{42} can be placed in R^*_{42} via gR. On the first 22 passes through D10, DL is set to "1" since a partial remainder is double length during this period. If a $23^{\underline{rd}}$ pass occurs, it is because the quotient did not appear normalized in R (i.e., nr = 0) after the 22nd pass through D11. On the $23^{\underline{rd}}$ pass es = -1 (i.e., -1 is in E), so DL is set to "0". This prevents the sign bits of the quotient from being placed in S_{43} and S_{44} during the $23^{\underline{rd}}$ pass through D11. If these bits are zeros, they could do no harm, but if they are units they could affect the quotient round-off. By setting DL to "0" during the $23^{\underline{rd}}$ pass through D10, it is guaranteed that zeros are placed in S_{43} and S_{44}.

The count in E is placed in EA during D10. Since -2sD is selected and C is set to "1", d = ea - 1 appears at the output of the D-adder.

During the first 22 passes through D11, the new partial remainder at the output of the A-adder and in the most significant end of Q is shifted left into SR. The quotient contained by the remaining bits of Q is likewise shifted left into R. The bit in Q^*_{42} is lost as explained in section 3.1.17. The new base 4 quotient digit (B_{42}, ρ_{43}, ρ_{44},) is gated into R^*_{42}, R_{43}, and R_{44}. The count, d = ea - 1, is gated into E. The output of the exponent decoder is also gated into ED as usual.

On the $23^{\underline{rd}}$ pass through D11, zeros are gated into S_{43} and S_{44} regardless of the contents of Q_{-1} and Q_0 since DL was set to "0" during the $23^{\underline{rd}}$ pass through D10. The reason for this is given above.

During all passes through D11 the ρMsS logic at the output of the A-adder determines the setting of the MsS mechanism as 2MsS, 0MsS or $\overline{2\text{Ms}}$S. It sets G'_1 and H'_1 accordingly. Note that changing the contents of the G'_1 and H'_1 will not affect the outputs of the quotient digit recoder (i.e., β_{42}, ρ_{43}, and ρ_{44}). Note also that the quotient "bit", $y_{45}(y_{47})$, placed in G'_1 and H'_1 during the $22^{\underline{nd}}$ ($23^{\underline{rd}}$) pass through D11 is not used to form the fractional quotient (y_{45} is used if nr = 0 after the $22^{\underline{nd}}$ pass through D11). Instead, this "bit" is used in conjunction with the sign of the resulting partial remainder (as placed in A during D12) to determine the quotient round-off.

After 22 passes through D11, es = -1 but the fractional quotient in R may or may not appear normalized. If nr = 1, control passes directly to D12. If nr = 0, the $23^{\underline{rd}}$ pass through D10 and D11 is executed. At the end of the $23^{\underline{rd}}$ pass through D11, the quotient in R must appear normalized, so nr = 1 and es = -2. Control then passes to D12.

In D12, the partial remainder, beyond the one which is corrected for

quotient round-off and retained as the remainder, is placed in A. The normalized quotient in R is placed in Q. KgA is selected so that the true sign of the excess partial remainder in A appears as α_{-1}. The unmodified quotient exponent, eq = eD - ed, is placed in EA. Note that the MsA request does not occur so the contents of G_1' and H_1' are not transferred to G_1 and H_1.

In D13, the output of the carry-borrow logic is gated into CB as shown on SEC (D-1527). Since Δ_2 is still set to "1", cb is logically equivalent to γ. There is no provision for remembering γ, so the carry-borrow logic and the CB memory element are used instead. This use necessitates the KgA request in D12. The CB output is used in conjunction with the quotient digit contained by G_1' and H_1' to determine the quotient round-off.

In order to place the unshifted remainder back in A, OMsS is requested and ∇ is set to "1". As discussed in section 3.1.2.6 and shown on HAS (D-1522), ∂ = 1 permits $s\sigma_{-3}$ and $s\sigma_{-2}$ to be gated into A_{-1} and A_0 via gA while σ^*_{-2} is placed in A^*_0. Since OMsS is selected, $s\sigma_{-3}$ and $s\sigma_{-2}$ represent the assimilated value of s_{-3}, s_{-2}, and s^*_{-2}.

The unmodified quotient exponent, eq = eD - ed, is in EA at the start of D13. This exponent may or may not be the true exponent of the fractional quotient in Q at this point. The initial subnormalization of D requires that a unit be added to eq. However, if an extra cycle of the divide loop is required to achieve a normalized quotient, a unit must be subtracted from eq + 1 leaving eq. Therefore, eq is the true exponent of the fractional quotient in Q if an extra cycle of the divide loop is required. In this case, the count placed in E during the final pass through D11 was -2. Hence, the ES = -2 F-element in the ED register is set to "1". This sets C to "0" during D13. Since D13 requests OsD, the quotient exponent appears at the output of the D-adder. If an extra cycle of the divide loop is not required, the ES = 2 F-element is set

to "0" during the final pass through D11. Hence C is set to "1" during D13 and $eq' = dq + 1$ as required.

During D14, the remainder is corrected in accordance with the quotient round-off which occurs in D17. The uncorrected remainder multiplied by 4 appears in S at the start of D14. It is right shifted into A during D14. Note that this depends on ∇ being set to "1". Note also that the bits in S_{43} and S_{44} are lost. If an extra cycle of the divide loop was required (i.e., if $1/4 \leq |D/d| \leq 1$), these bits are both zero. If an extra cycle was not required, (i.e., if $1 < |D/d| \leq 4$), these bits are the two least significant bits of D. If either of these are nonzero, $(d4^{ed})(q4^{eq}) + R_r \cdot 4^{eR}$ will not equal $D4^{eD}$. Δ_2 is set to "0" since the output of the carry-borrow logic was used during D13.

As discussed earlier in this section, the quotient round-off is -1.2^{-44} if $m_{-1} \oplus \alpha_{-1} = cb = 1$ and $2y_{45}(2y_{47}) = -2$, the round-off is $+1.2^{-44}$ if $m_{-1} \oplus \alpha_{-1} = cb = 0$ and $2y_{45}(2y_{47}) = +2$, and the round-off is zero otherwise. The last quotient "bit" predicted, y_{45} or y_{47}, is held in G_1' and H_1' but is not inserted in the fractional quotient. If $y_{45}(y_{47}) = -1$, $g_1' = 1$ and $h_1' = 0$. If $y_{45}(y_{47}) = +1$, $g_1' = h_1' = 0$. If $y_{45}(y_{47}) = 0$, $g_1' = 0$ or 1 and $h_1' = 1$.

The fractional divisor is added to the remainder in A to compensate for a negative quotient round-off. It is subtracted from the remainder in A to compensate for a positive quotient round-off. Zero is added to the remainder if the quotient round-off is zero. Hence, MsA is requested if $g_1'\overline{h_1'}(cb) = 1$; $\overline{\text{MsA}}$ is requested if $\overline{g_1'}\,\overline{h_1'}(\overline{cb}) = 1$; and By MsA (leaving KgA as the setting) is requested if $h_1' \vee (g_1' \oplus cb) = 1$.

The exponent of the fractional quotient in Q, eq, or eq' is gated into E during D14. Note that C is still set to "0" if an extra cycle of the divide loop was required but is set to "1" otherwise. Note also that the ESOV F-element in the ED register is set to "1" if eq or $eq' > 127$.

In D15, the fractional remainder corrected for quotient round-off, R_r,

appears as α and is shifted left into S. OV is set to "1" if esov = 1 and is bypassed otherwise. The divisor exponent is **added** to eq or eq' by requesting EMsD. Thus d = eq + c + ed appears at the output of the D-adder. Note that d = eD - ed + c + ed = eD + c. Let the remainder exponent be denoted by eR. If an extra cycle of the divide loop was required, eR = eD' - 23 = eD - 22. If an extra cycle of the divide loop was not required, eR = eD' - 22 = eD - 21. Since C is set to "0" in the first case and to "1" in the second, it is clear that eR = eD + c - 22. This is accomplished during D17. In D15, eD + c appears at the output of the D-adder.

During D16, 4R in S is shifted right into A and appears in assimilated form at the output of the A-adder. Its magnitude is less that or equal to half of the magnitude of the divisor. In the EAU eq + c is placed in ES. DL is set to "0" to prevent the (K) sequence from destroying the sign bits of the fractional remainder in R in the event q or q' = +1. This request is made during D16 simply because it is convenient from the standpoint of physical layout. From a strictly logical standpoint, it could have been made during D17, D18, D19, or D20.

During D17, the remainder, R_r, is placed in R while the quotient is placed in S. If $\overline{g_1'}\overline{h_1'}(\overline{cb}) = 1$, CS and By $M\overline{M}$ is requested to add a unit to q or q' in the $44^{\underline{th}}$ position. If $g_1'\overline{h_1'}(cb) = 1$, $\overline{CS}$ and $M\overline{M}sS$ is requested to subtract a unit from q or q' in the $44^{\underline{th}}$ position. Otherwise, zero is added to q or q' by requesting By CS and By $M\overline{M}$. The rounded quotient thus appears at the output of the S-adder. ∇ is set to "0" since the right shift path from S to A is no longer needed. In the EAU, C is set to "0" and -22sD is requested while eq + c in ES is transferred to EA. Thus eR = eq + c - 22 appears at the output of the D-adder.

During D18, the rounded quotient is placed in A (it may be +1) while Q is cleared to zero. Note that the quotient exponent, eq + c = eq or eq', is

held in E while eR is still appearing at the output of the D-adder.

In D19, the MsS selector mechanism is cleared to zero as usual following a possible $\overline{M}$MsS or CS request. Zero is selected to SR. If the remainder in R is zero, RZ = 1. If eR $\leq$ -64, then (d $\leq$ -64) = 1. In either case, R is cleared to zero via a gR while -64 is placed in ES. Otherwise, the remainder is retained in R and eR is gated into ES. Thus the remainder corrected for quotient round-off appears in R and ES at the end of any divide order. The (K) sequence cannot affect it since DL is set to "0". The SRM order discussed in section 5.7 must occur immediately after the divide order in the DC sequencing if the remainder is to be retained.

A null step, D20, separates D19 from K1 since the gR reply is used in both D19 and K1.